Deciding Properties of Automatic Sequences

by

Luke Schaeffer

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, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

In this thesis, we show that several natural questions about automatic sequences can be expressed as logical predicates and then decided mechanically. We extend known results in this area to broader classes of sequences (e.g., paperfolding words), introduce new operations that extend the space of possible queries, and show how to process the results.

We begin with the fundamental concepts and problems related to automatic sequences, and the corresponding numeration systems. Building on that foundation, we discuss the general logical framework that formalizes the questions we can mechanically answer. We start with a first-order logical theory, and then extend it with additional predicates and operations. Then we explain a slightly different technique that works on a monadic second-order theory, but show that it is ultimately subsumed by an extension of the first-order theory.

Next, we give two applications: critical exponent and paperfolding words. In the critical exponent example, we mechanically construct an automaton that describes a set of rational numbers related to a given automatic sequence. Then we give a polynomial-time algorithm to compute the supremum of this rational set, allowing us to compute the critical exponent and many similar quantities. In the paperfolding example, we extend our mechanical procedure to the paperfolding words, an uncountably infinite collection of infinite words.

In the following chapter, we address abelian and additive problems on automatic sequences. We give an example of a natural predicate which is provably inexpressible in our first-order theory, and discuss alternate methods for solving abelian and additive problems on automatic sequences.

We close with a chapter of open problems, drawn from the earlier chapters.

Acknowledgements

I would like to thank Jeffrey Shallit for supervising my research, and for his assistance preparing this thesis. I would also like to thank the readers, Shai Ben-David and Jason Bell, for agreeing to read this thesis. Finally, I would like to thank my friends and family for their encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The goal of this thesis is to explain and develop a procedure for mechanically deciding questions about automatic sequences, which we state as predicates in first-order logic. After explaining the necessary background for automatic sequences, we present the basic decidability result in Theorem 3.1. This is not a new result, but we will apply it and extend it many different ways.

First, we extend our logical theory with a new operation in Section 3.4. This allows us to apply a finite automaton to compare two subwords of a sequence, solving an open problem. It is also general enough to apply to other extensions in later chapters.

Second, we give an algorithm (Theorem 4.20) that efficiently processes the output of our mechanical procedure, improving on an earlier result [49]. This algorithm allows us to answer questions about ratios and limits that cannot be expressed directly as predicates.

The third major result (Theorem 4.24) extends our decision procedure (and the first-order logical theory) to an uncountably infinite family of sequences, called paperfolding words. This answers an open problem about paperfolding words, and gives proves a number of other interesting new properties.

Finally, we address abelian and additive problems in automatic sequences, which are difficult to state as predicates. In fact, we show that a specific, natural query about abelian powers is inexpressible in Theorem 5.5. On the other hand, we show the existence of an additive cube-free word in Theorem 5.12, based on the author's work in [15].

Let us first introduce a number of basic concepts and definitions. We begin with finite words, languages, monoids, morphisms, automata and ω-automata. The remainder of this chapter covers notation and definitions which make up the foundations of the theory of automatic sequences.

1.2 Words and Languages

Let Σ^* denote the set of (finite) words over a finite alphabet Σ. Unless stated otherwise, we will use the symbols Σ, Δ and Γ to represent finite alphabets. We write $\varepsilon \in \Sigma^*$ for the empty word. Let $|x|$ denote the length of a word $x \in \Sigma^*$, and $|x|_a$ denote the number of occurrences of a symbol $a \in \Sigma$ in the word x. Let Σ^ω denote the set of one-sided right-infinite words over Σ.

Let $w \in \Sigma^*$ be a word and suppose $i \geq 0$ is an integer. We write $w[i]$ to denote the symbol in w at position i. We take the first symbol in w to be at position 0, so

$$w = w[0]w[1]w[2]\cdots w[n-1]$$

where $n = |w|$. A *subword* of w is a contiguous block of symbols in w. We write $w[i..j]$ for the subword $w[i]w[i+1]\cdots w[j-1]w[j]$. In the case of infinite words, we allow ∞ as a position so that we may write $w[i..\infty]$ for a suffix of w.

A *language over* Σ is a subset of Σ^*. The traditional set operations of union $(A \cup B)$, intersection $(A \cap B)$, set difference $(A \backslash B)$, symmetric difference $(A \oplus B)$ and complement $(\Sigma^* \backslash A)$ are common operations on languages $A, B \subseteq \Sigma^*$. In addition, we have the following language operations for languages $A, B, C \subseteq \Sigma^*$.

- The *concatenation* of two languages, denoted AB, is defined to be

  $$AB := \{ab : a \in A, b \in B\}.$$

  Note that $A(BC) = (AB)C$, so concatenation is associative, and $A\{\varepsilon\} = A = \{\varepsilon\}A$. We define powers of a language, A^n, where

  $$A^n := \begin{cases} \{\varepsilon\}, & \text{if } n = 0; \\ A^{n-1}A, & \text{if } n > 0. \end{cases}$$

- Let A^* be the *Kleene star* of A, defined by

$$A^* := \bigcup_{n \geq 0} A^n.$$

  The *Kleene plus* is similarly defined as $A^+ := \bigcup_{n \geq 1} A^n$.

- The *quotient of A by B* is the language

$$A/B := \{x \in \Sigma^* : \text{there exists } y \in B \text{ such that } xy \in A\}.$$

  Note that $(AB)/B \supseteq A$, but in general $(AB)/B \neq A$.

Definition 1.1. We give a recursive definition for the collection of regular languages over an alphabet Σ. A language $L \subseteq \Sigma^*$ is *regular* if

1. L is finite, or
2. there exist regular languages $A, B \subseteq \Sigma^*$ such that $L = AB$, or
3. there exist regular languages $A, B \subseteq \Sigma^*$ such that $L = A \cup B$, or
4. there exists a regular language $A \subseteq \Sigma^*$ such that $L = A^*$.

In other words, close the set of finite languages under concatenation, union and Kleene star to obtain the set of regular languages.

We note without proof that regular languages are closed under all the language operations we have described.

Theorem 1.2. *Let $A, B \subseteq \Sigma^*$ be regular languages. Then $A \cup B$, $A \cap B$, $A \backslash B$, $A \oplus B$, $\Sigma^* \backslash A$, AB, A^*, A^+, and A/B are regular languages.*

1.3 Monoids

Definition 1.3. A *monoid* is a triple, $(M, \cdot, 1)$ satisfying the following conditions:

1. M is a set, $\cdot : M \times M \to M$ is a binary operation on M, and $1 \in M$ is an element.
2. for all $x, y, z \in M$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$. That is, the operation is associative.
3. for all $x \in M$, $1 \cdot x = x = x \cdot 1$. In other words, 1 is a two-sided identity for the operation.

We say a monoid is *commutative* if it satisfies the additional condition

4. for all $x, y \in M$, $x \cdot y = y \cdot x$.

The *monoid of words over* Σ is $(\Sigma^*, \cdot, \varepsilon)$, the set of words Σ^* under concatenation, where ε is the identity element. Another common example is $(\mathbb{N}, +, 0)$, the set of nonnegative integers under addition. The set of all languages under concatenation is a monoid, since we have seen that $A(BC) = (AB)C$ for languages $A, B, C \in \Sigma^*$ and $A\{\varepsilon\} = A = \{\varepsilon\}A$. We can also construct monoids from sets or other monoids.

1. Suppose X and Y are sets. Let X^Y denote the set of functions from X to Y. Then $(X^X, \cdot, \mathrm{id})$ is a monoid, where $\mathrm{id}\colon X \to X$ is the identity function and the monoid operation is $(fg)(x) := g(f(x))$ (i.e., the opposite of function composition).

2. Let $(M, +, 0)$ be a monoid, and let X be a set. It is clear that $(M^X, \oplus, \mathbf{0})$ is a monoid, where $(f \oplus g)(x) := f(x) + g(x)$ for all $x \in X$, $f, g \in M^X$, and $\mathbf{0}$ is the zero function.

When the binary operation and identity of a monoid $(M, \cdot, 1)$ are understood, we will refer to the monoid by M, its ground set.

Like many other mathematical objects, there exist maps between monoids that preserve the monoid structure.

Definition 1.4. A *monoid homomorphism* is a function $\varphi\colon X \to Y$ from a monoid $(X, \cdot_X, 1_X)$ to a monoid $(Y, \cdot_Y, 1_Y)$ such that $f(1_X) = 1_Y$ and

$$f(a \cdot_X b) = f(a) \cdot_Y f(b)$$

for all $a, b \in X$.

A *morphism* is a monoid homomorphism where the domain and/or codomain are monoids of words under concatenation.

For example, the map $\ell\colon \Sigma^* \to \mathbb{N}$ such that $\ell(x) := |x|$ is an example of a monoid homomorphism because $\ell(\varepsilon) = |\varepsilon| = 0$ and

$$\ell(xy) = |xy| = |x| + |y| = \ell(x) + \ell(y).$$

Suppose $\varphi\colon \Sigma^* \to M$ is a monoid homomorphism. Given the values $\varphi(a)$ for all $a \in \Sigma$, we deduce that

$$\varphi(a_1 \cdots a_n) = \varphi(a_1) \cdots \varphi(a_n).$$

Therefore we can specify a morphism φ by giving its value on each $a \in \Sigma$. The following theorem gives a converse: any assignment $\Sigma \to M$ can be extended to a homomorphism.

Theorem 1.5. *Let M be a monoid, and let Σ be a finite alphabet. Suppose $f\colon \Sigma \to M$ is an arbitrary function. There exists a unique morphism $\hat{f}\colon \Sigma^* \to M$ such that $\hat{f}(a) = f(a)$ for all $a \in \Sigma$.*

For example, consider the map $\Sigma \to \mathbb{N}^\Sigma$ that sends $a \in \Sigma$ to the function $\delta_a \colon \Sigma \to \mathbb{N}$ where

$$\delta_a(b) := \begin{cases} 0, & \text{if } a \neq b; \\ 1, & \text{if } a = b. \end{cases}$$

This extends to the monoid homomorphism $\psi \colon \Sigma^* \to \mathbb{N}^\Sigma$ such that $\psi(x)(a) = |x|_a$. This homomorphism is called the *Parikh map* after its use by Parikh in [39].

We can also use Theorem 1.5 to succinctly define morphisms. For example, consider the morphism $h \colon \{1,2,3,4\}^* \to \{\mathtt{i}, \mathtt{m}, \mathtt{p}, \mathtt{s}\}^*$ uniquely defined by the following data.

$$\begin{aligned} h(1) &= \mathtt{m} \\ h(2) &= \mathtt{iss} \\ h(3) &= \mathtt{pp} \\ h(4) &= \mathtt{i} \\ h(5) &= \varepsilon \end{aligned}$$

Then we have

$$h(1224534) = \mathtt{mississippi}.$$

1.4 Finite Automata

In general terms, an *automaton* is a machine that reads an *input word* from left to right, and produces some *output*. The machine has a *state* between each symbol in the word, and updates the state as it reads symbols from the input. The prototypical example is the *deterministic finite automaton* (DFA), defined below.

Definition 1.6. A *deterministic finite automaton (DFA)* T is a 5-tuple $(\Sigma, Q, \delta, q_0, F)$, where

- Σ is a finite alphabet,
- Q is the (finite) set of *states*,
- $\delta \colon Q \times \Sigma \to Q$ is the *transition function*,
- $q_0 \in Q$ is the *initial state*, and
- $F \subseteq Q$ is a set of *final states*.

We extend δ to $\delta^* \colon Q \times \Sigma^* \to Q$ such that

$$\delta^*(q, w) = \delta(\delta(\cdots \delta(\delta(q, w[0]), w[1]) \cdots, w[n-2]), w[n-1])$$

for $w = w[0..n-1] \in \Sigma^*$.

Given a word $w \in \Sigma^*$, we say T *accepts* w if $\delta^*(q_0, w) \in F$, otherwise we say T *rejects* w. The *language recognized by* T, denoted $L(T)$, is the set of words in Σ^* accepted by T.

DFAs are interesting as a model of computation because they accept precisely the class regular languages.

Theorem 1.7. *A language $L \subseteq \Sigma^*$ is accepted by a DFA if and only if L is regular.*

All of the closure properties in Theorem 1.2 can be constructively proved on DFAs.

We state the *pumping lemma* for DFAs without proof for future use.

Lemma 1.8. *Let $M = (\Sigma, Q, \delta, q_0, F)$ be a DFA recognizing the language $L = L(M) \subseteq \Sigma^*$. There exists a constant ℓ (depending on M, especially $|Q|$) such that for any word $w \in L$ of length $|w| \geq \ell$, we can factor w into xyz for $x, y, z \in \Sigma^*$ such that $xy^iz \in L$ for all $i \geq 0$ and $1 \leq |y| \leq \ell$.*

Furthermore, we can pick a length ℓ window (i.e., subword) in w, such that when we factor $w = xyz$, the subword y is contained in the window.

Given an input word, the output of a DFA is a boolean flag indicating whether to accept or reject the input. Deterministic finite automata *with output* (DFAOs) naturally generalize DFAs by allowing outputs in a finite set.

Definition 1.9. A *deterministic finite automaton with output (DFAO)* T is a 6-tuple $(\Sigma, Q, \delta, q_0, \Gamma, \gamma)$, where

- Σ is a finite alphabet,
- Q is the (finite) set of *states*,
- $\delta\colon Q \times \Sigma \to Q$ is the *transition function*,
- $q_0 \in Q$ is the *initial state*,
- Γ is an output alphabet (not necessarily finite),
- $\gamma\colon Q \to \Gamma$ is the output map.

As with DFAs, we extend δ to $\delta^*\colon Q \times \Sigma^* \to Q$.

The output of T on a string $w \in \Sigma^*$, denoted $T(w) \in \Gamma$, is defined as

$$T(w) := \gamma(\delta^*(q_0, w)).$$

When we want to define a DFAO without naming the internals (state set, initial state and output map), we write $T\colon \Sigma^* \to \Gamma$.

Definition 1.10. We say a function $f\colon \Sigma^* \to \Gamma$ is *automatic* if there exists a DFAO $T\colon \Sigma^* \to \Gamma$ such that $T(w) = f(w)$ for all $w \in \Sigma^*$.

We are primarily interested in automatic functions where the input is a string of digits representing a number. For instance, for any fixed $m \geq 2$, there is a finite automaton that computes n mod m given the binary expansion of n. We will also use automatic functions to define automatic sequences in Chapter 2.

DFAOs are closed under products, post-composition with functions and pre-composition with morphisms. Consequently, we have the following closure properties for automatic functions, which we state without proof.

Theorem 1.11. *Let $f_1\colon \Sigma^* \to \Gamma_1$ and $f_2\colon \Sigma^* \to \Gamma_2$ be automatic functions. Then the product function $(f_1 \times f_2)\colon \Sigma^* \to \Gamma_1 \times \Gamma_2$, where $(f_1 \times f_2)(x) = (f_1(x), f_2(x))$ for all $x \in \Sigma^*$, is automatic.*

Theorem 1.12. *Let $f\colon \Delta^* \to \Sigma^*$ be a morphism, let $g\colon \Sigma^* \to \Gamma$ be an automatic function, and let $h\colon \Gamma \to \Theta$ be any function. Then the composition $h \circ g \circ f\colon \Delta^* \to \Theta$ is an automatic function.*

We can also assemble DFAOs from a collection of regular languages, or vice versa. As a corollary, we can represent an automatic function as a collection of automatic functions into $\{0, 1\}$. This will be useful when we use automata to represent logical formulas in Chapter 3, since logical predicates are inherently boolean.

Theorem 1.13. *Let Σ, Γ be finite alphabets and suppose $f\colon \Sigma^* \to \Gamma$ is a function. Then f is automatic if and only if the language*

$$L_a := \{w \in \Sigma^* : f(w) = a\}$$

is regular for each $a \in \Gamma$.

Proof. Suppose f is an automatic function. For each $a \in \Gamma$, define a function $I_a\colon \Gamma \to \{0, 1\}$ where

$$I_a(b) = \begin{cases} 0, & \text{if } a \neq b; \\ 1, & \text{if } a = b. \end{cases}$$

Then $I_a\colon f$ is an automatic function, and the corresponding DFAO outputs 1 for exactly the words in L_a. Hence, L_a is regular.

Conversely, suppose each L_a is regular. There exist automatic functions $g_a\colon \Sigma^* \to \{0,1\}$ such that

$$g_a(w) = \begin{cases} 0, & \text{if } w \notin L_a; \\ 1, & \text{if } w \in L_a. \end{cases}$$

Then the product of these functions, $g\colon \Sigma^* \to \{0,1\}^\Gamma$, is automatic. It is not hard to see that for all $w \in \Sigma^*$, $g(w) \in \{0,1\}^\Gamma$ is a tuple of zeros with a 1 in coordinate $f(w)$. Hence, f is the composition of g with an appropriate function $h\colon \{0,1\}^\Gamma \to \Gamma$, which is automatic by Theorem 1.12. □

We also have an algebraic characterization of automatic functions in terms of monoids.

Theorem 1.14. *Let $f\colon \Sigma^* \to \Gamma$ be a function. Then f is automatic if and only if there exist a finite monoid M, morphism $\mu\colon \Sigma^* \to M$ and function $t\colon M \to \Gamma$ such that $f = t \circ \mu$.*

Proof. If f is automatic then we have a corresponding DFAO $T = (\Sigma, Q, \delta, q_0, \Gamma, \gamma)$ that computes the function. We take

- $M = Q^Q$,
- $\mu\colon \Sigma^* \to Q^Q$ where $\mu(x) := q \mapsto \delta^*(q, x)$, and
- $t\colon Q^Q \to \Gamma$ where $t(g) := \gamma(g(q_0))$.

Then

$$f(x) = \gamma(\delta^*(q_0, x)) = \gamma\Big((q \mapsto \delta^*(q, x))\,(q_0)\Big) = t(\mu(x))$$

for all $x \in \Sigma^*$, so $f = t \circ \mu$.

In the other direction, suppose we are given a finite monoid M, morphism $\mu\colon \Sigma^* \to M$ and function $t\colon M \to \Gamma$ such that $f = t \circ \mu$. Then we define $T = (\Sigma, M, \delta, q_0, \Gamma, t)$ where

- the set of states is M,
- the set of output associated with a state $q \in M$ is $t(q)$,
- the initial state is $q_0 = 1 \in M$ where 1 is the identity element in M,
- and $\delta(q, a) := q\mu(a)$ for all $q \in M$ and $a \in \Sigma$.

Then observe that

$$\begin{aligned} \delta^*(q, w) &= \delta(\delta(\cdots\delta(\delta(q, w[0]), w[1])\cdots, w[n-2]), w[n-1]) \\ &= ((\cdots((q\mu(w[0]))\mu(w[1]))\cdots\mu(w[n-2]))\mu(w[n-1])) \\ &= q\mu(w[0])\mu(w[1])\cdots\mu(w[n-2])\mu(w[n-1]) \\ &= q\mu(w) \end{aligned}$$

for all $q \in M$ and $w = w[0..n-1] \in \Sigma^*$. In particular,

$$t(\delta^*(q_0, w)) = t(1 \cdot \mu(w)) = (t \circ \mu)(w) = f(w)$$

for all w, so $T(w) = f(w)$ as desired. □

1.4.1 ω-Languages and ω-Automata

In this section, we discuss the generalizations of languages and automata from finite words to infinite words. An *ω-language* is a subset of Σ^ω, the set of (right) infinite words over Σ, and there is a corresponding generalization of regular languages.

Definition 1.15. A ω-language $L \subseteq \Sigma^\omega$ is *ω-regular* if it is of the form

- $A^\omega := \{a_1 a_2 a_3 \cdots : a_1, a_2, a_3, \ldots \in A\}$ for A a regular language,
- $AB := \{ab : a \in A, b \in B\}$ for A a regular language and B an ω-regular language, or
- $A \cup B$ for ω-regular languages A, B.

In addition to being closed under A^ω, AB and $A \cup B$, ω-regular languages are also closed under intersection ($A \cap B$), complement ($\overline{A}$), morphism ($\varphi(A)$) and inverse morphism ($\varphi^{-1}(A)$). See [40, Chapter 1] for proofs of these closure properties and further discussion of ω-regular languages (in their terminology, ω-rational sets).

We can also characterize ω-regular languages with *ω-automata*. There are many kinds of ω-automata, but we will only describe Büchi automata and Muller automata. Büchi automata were introduced by Büchi in [12] for the purpose of deciding logical statements.

Definition 1.16. A *Büchi automaton* T is a 5-tuple $(\Sigma, Q, \delta, q_0, F)$ where

- Σ is a finite alphabet,
- Q is a (finite) set of *states*,
- $\delta \colon Q \times \Sigma \to 2^Q$ is a non-deterministic *transition function*,
- $q_0 \in Q$ is an *initial state*,
- $F \subseteq Q$ is a set of final states.

Then T accepts a word $w \in \Sigma^\omega$ if it labels an infinite walk through the transition digraph that visits some state in F infinitely many times.

In other words, a Büchi automaton has the same elements as an NFA, but with a different acceptance condition. As an example, consider the Büchi automaton in Figure 1.1, which accepts the language

$$L = \{w \in \{0,1\}^\omega : w \text{ contains infinitely many 1s}\}.$$

The complement language,

$$\overline{L} = \{w \in \{0,1\}^\omega : w \text{ contains finitely many 1s}\}$$

is also accepted by a Büchi automaton, shown in Figure 1.2.

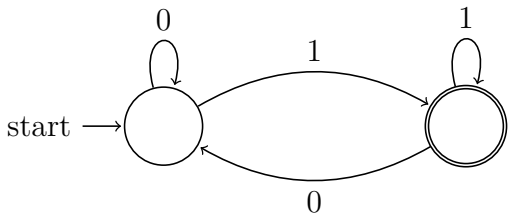


Figure 1.1: Büchi automaton accepting words with infinitely many 1s.

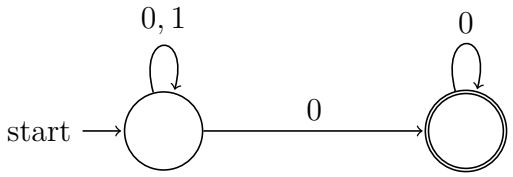


Figure 1.2: Büchi automaton accepting words with finitely many 1s.

One downside of Büchi automata is that they are necessarily nondeterministic. For instance, there is no deterministic Büchi automaton for the ω-language of words with finitely many 1s. Muller automata, on the other hand, are deterministic and use a slightly different acceptance condition.

Definition 1.17. A *Muller automaton* is a 5-tuple $T = (\Sigma, Q, \delta, q_0, \mathbf{F})$ where

- Σ is a finite alphabet,
- Q is the (finite) set of *states*,
- $\delta\colon Q \times \Sigma \to Q$ is the *transition function*,
- $q_0 \in Q$ is the *initial state*, and
- $\mathbf{F} \subseteq 2^Q$ is a collection of subsets of Q.

Define $\delta^*\colon Q \times \Sigma^* \to Q$ by extending δ, as we did for DFAs. We say T accepts an infinite word $w \in \Sigma^\omega$ if the set

$$\{q \in Q : \delta^*(q_0, w[0..n-1]) = q \text{ for infinitely many } n \in \mathbb{N}\}$$

belongs to $\mathbf{F}$.

The following result relates ω-automata to ω-regular languages.

Theorem 1.18. *Let $L \subseteq \Sigma^\omega$ be an ω-language. The following are equivalent*

- *L is ω-regular.*
- *L is recognized by a Büchi automaton.*
- *L is recognized by a Muller automaton.*

Proof. See [40, Chapter 1]. □

Furthermore, the closure properties of ω-regular languages correspond to constructions on ω-automata.

1.4.2 Automata with Multiple Inputs

Sometimes we will need automata that take a fixed number of inputs, say $d \in \mathbb{N}$, over an alphabet Σ (or occasionally over heterogeneous alphabets). For ω-automata, this is trivial; we simply use an ω-automaton over the alphabet Σ^d. As a notational convenience, if $a_1, a_2, \ldots, a_m \in \Sigma^\omega$ then we write $(a_1, \ldots, a_m)$ for the infinite word

$$(a_1[0], \ldots, a_m[0])(a_1[1], \ldots, a_m[1])(a_1[2], \ldots, a_m[2]) \cdots \in (\Sigma \times \Sigma)^\omega.$$

In other words,$(\Sigma^\omega)^d$ is isomorphic to $(\Sigma^d)^\omega$, and to make notation simpler, we may pretend we are in one when we are in the other. For example, given ω-languages $A \subseteq \Sigma^\omega$ and $B \subseteq \Sigma^\omega$, we have

$$\begin{aligned} A \times B &= \{(a, b) : a \in A, b \in B\} \\ &= \{(a, b) \in (\Sigma \times \Sigma)^\omega : a \in A\} \cap \{(a, b) \in (\Sigma \times \Sigma)^\omega : b \in A\} \\ &= h_1^{-1}(A) \cap h_2^{-1}(B) \end{aligned}$$

where $h_1, h_2\colon (\Sigma \times \Sigma)^* \to \Sigma^*$ are morphisms such that $h_1(a, b) = a$ and $h_2(a, b) = b$. Hence, $A \times B$ is ω-regular.

For automata on finite words, the situation is more complicated because the inputs may have different lengths, and hence $\Sigma^* \times \Sigma^*$ is not isomorphic to $(\Sigma \times \Sigma)^*$. Our solution is to pad all input words to the same length with a new symbol, $\square$, and work over the alphabet $(\Sigma \cup \{\square\})^d$. That is, given $x_1, x_2, \ldots, x_d \in \Sigma^*$, we find strings $y_1, y_2, \ldots, y_d \in (\Sigma \cup \{\square\})^*$ such that $y_i = x_i \square^{e_i}$ for all i, and $|y_1| = |y_2| = \cdots = |y_d| = n$. Then we feed

$$(y_1[1], \ldots, y_d[1])(y_1[1], \ldots, y_d[2]) \cdots (y_1[n], \ldots, y_d[n])$$

into the automaton. We will sometimes write $(x_1, \ldots, x_d)$ for the string

$$(y_1[1], \ldots, y_d[1]) \cdots (y_1[n], \ldots, y_d[n]) \in (\Sigma \cup \{\square\})^*$$

since it is cumbersome to write the latter.

Now for some terminology. We say $\square$ is the *padding symbol*, and call the appended $\square$s *padding*. We also extend a number of language and automata-related terms to multiple inputs.

- We say a subset of $(\Sigma^*)^d$ is a *language*. Similarly, a subset of $(\Sigma^\omega)^d$ is an *ω-language*.
- A language $L \subseteq (\Sigma^*)^d$ is *regular* if some DFA over $(\Sigma \cup \{\square\})^d$ recognizes tuples $(x_1, \ldots, x_d)$ in L.
- A function $f\colon (\Sigma^*)^d \to \Gamma$ is *automatic* if some DFAO $T\colon (\Sigma \cup \{\square\})^d \to \Gamma$ outputs $f(x_1, \ldots, x_d)$ given $x_1, \ldots, x_d$ as input.

For instance, the set

$$\{(x, y) \in \Sigma^* \times \Sigma^* : x \text{ is a prefix of } y\}$$

is an example of a regular language over multiple inputs, since it is recognized by the automaton in Figure 1.3.

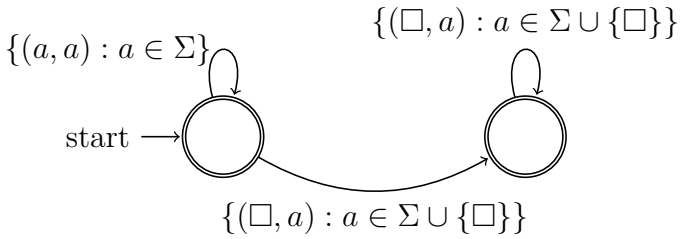


Figure 1.3: Automaton deciding if one input is a prefix of the other.

For our applications, most inputs will be strings of digits intended to represent numbers. This raises an important issue about padding. In most cases, an extra padding symbol (e.g., $\square$) is unnecessary because we can already pad numbers with leading zeros. That is, $153, 0153, 00153, \ldots$ all represent the number one hundred and fifty-three, just as $\texttt{ab}, \texttt{ab}\square, \texttt{ab}\square\square, \ldots$ are all representations for $\texttt{ab}$ with different amounts of padding. We will assume that

- numbers are padded with 0s instead of $\square$s, and
- the padding zeros are added to the most significant end of the number, so that the least significant digits line up,

unless stated otherwise. We discuss the problem again when we discuss numeration systems in Section 2.2.

Another minor issue is the definition of an automaton with zero inputs. This comes up in Chapter 3 (see Theorem 3.1), when we convert logical formulas into automata. The number of inputs to the automaton is the number of free variables in the formula. Hence, if there are no free variables then the automaton has zero inputs. To make sense of this case, we define an automaton with zero variables as an automaton over Σ^0, and we define Σ^0 to be a singleton set, say $\{\square\}$. For our purposes, an automaton with zero inputs will either accept all words or reject all words, according to whether the logical formula is true or false.

1.5 Morphic Words

Define $t\colon \{0,1\}^* \to \{0,1\}^*$, the *Thue-Morse morphism*, where $t(0) = 01$ and $t(1) = 10$. Consider the sequence $(t^i(0))_{i=0}^{\infty}$,

$$\begin{aligned} t^0(0) &= 0 \\ t^1(0) &= 01 \\ t^2(0) &= 0110 \\ t^3(0) &= 01101001 \\ t^4(0) &= 0110100110010110 \\ &\vdots \end{aligned}$$

Note that each term is a prefix of the next. We can define an infinite word, $\mathbf{t} \in \{0,1\}^\omega$, called the *Thue-Morse word* [6, p. 15], which is the limit of these prefixes.

$$\mathbf{t} = 0110100110010110100101100110100 1 \cdots$$

That is, $\mathbf{t}$ is the unique word such that $t^n(0)$ is a prefix of $\mathbf{t}$ for all n. Observe that as a consequence, $\mathbf{t}$ is a fixed point of t (i.e., $\mathbf{t} = t(\mathbf{t})$). In general, we use the fixed point property to define pure morphic words.

Definition 1.19. We say an infinite word $w \in \Sigma^\omega$ is *pure morphic* if there exists a non-erasing morphism $\varphi\colon \Sigma^* \to \Sigma^*$ such that $\varphi(w) = w$ and $\varphi(w[0]) = w[0..n]$ for some $n \geq 1$.

Suppose $x \in \Gamma^\omega$ is an infinite word. We say x is *morphic* if it is the image of some pure morphic word $y \in \Sigma^\omega$ under a coding $h\colon \Sigma \to \Gamma$.

Let us give a few examples of pure morphic and morphic sequences.

- The Thue-Morse word, defined above, is pure morphic.
- The Fibonacci word [6, p. 212],

  $$\mathbf{f} = 01001010010010100 1010\cdots$$

  is defined as the fixed point of the morphism $0 \mapsto 01, 1 \mapsto 0$.
- The period-doubling sequence [6, Example 6.3.4, p. 176],

  $$\mathbf{d} = 10111010101110\cdots$$

  is the fixed point of $1 \mapsto 10, 0 \mapsto 11$.
- Consider the fixed point of

  $$\begin{aligned} a &\mapsto abcc \\ b &\mapsto b \\ c &\mapsto cc \end{aligned}$$

  under the coding $\mu\colon \{a,b,c\}^* \to \{0,1\}^*$ where $\mu(a) = 1, \mu(b) = 1, \mu(c) = 0$. This gives the word

  $$\begin{aligned} \mathbf{s} &= \mu(abccbccccbccccccbccccccccbccccccccccbccc\cdots) \\ &= 1100100001000000100000000100000000001000\cdots. \end{aligned}$$

  It turns out that this word can also be defined as the characteristic sequence of squares. That is,

  $$\mathbf{s}[n] = \begin{cases} 1, & \text{if } n \text{ is a square;} \\ 0, & \text{otherwise.} \end{cases}$$

- The Rudin-Shapiro sequence [6, Example 3.3.1, p. 78] is defined to be the fixed point of the morphism

$$\begin{aligned} a &\to ab \\ b &\to ac \\ c &\to db \\ d &\to dc \end{aligned}$$

under the coding $a, b \mapsto 0$, $c, d \mapsto 1$. That is,

$$\mathbf{r} = 00010010000111010001001011100 0 \cdots .$$

Alternatively, $\mathbf{r}[n]$ is the number of (possibly overlapping) occurrences of 11 in the binary representation of n.

- The Baum-Sweet sequence [6, Example 6.3.3, p. 176],

$$\mathbf{b} = 00100110101101100110111110110 1 \cdots ,$$

is the fixed point of

$$\begin{aligned} a &\to ab \\ b &\to cb \\ c &\to bd \\ d &\to dd \end{aligned}$$

under the coding $a, b \mapsto 0$ and $c, d \mapsto 1$.

- The Mephisto waltz word [6, Exercise 16, p. 25],

$$\mathbf{m} = 00100111000100111011011000 1 \cdots ,$$

is the fixed point of $0 \mapsto 001, 1 \mapsto 110$.

1.6 Properties of Infinite Words

To study infinite words, we have to study their finite subwords. This leads to a number of natural questions about subwords of infinite words, and corresponding terminology.

1.6.1 Subword Complexity

The simplest description of subwords is the subword complexity, which counts the number of subwords of each length.

Definition 1.20. Let $w \in \Sigma^\omega$ be an infinite word. The *subword complexity* of w is a function $\rho\colon \mathbb{N} \to \mathbb{N}$ such that $\rho(n)$ is the number of distinct subwords of length n in w.

The subword complexity is trivially a non-decreasing function, bounded above by $|\Sigma|^n$.

Theorem 1.21. *Let $w \in \Sigma^\omega$ be an infinite word with subword complexity $\rho\colon \mathbb{N} \to \mathbb{N}$. Then w is aperiodic if and only if $\rho(n) \geq n + 1$ for all n.*

Proof. See, e.g., [38] or [20]. □

On the other hand, we have the following bound for morphic words.

Theorem 1.22. *Let $w \in \Sigma^\omega$ be a morphic word with subword complexity $\rho\colon \mathbb{N} \to \mathbb{N}$. Then $\rho(n)$ is in $O(n^2)$.*

Proof. See [6, Corollary 10.4.9, p. 310]. □

1.6.2 Recurrence, Appearance and Condensation

Sometimes in the analysis of infinite words, it is useful to assume that every subword eventually occurs again. This leads to the definition of a recurrent word.

Definition 1.23. An infinite word w is *recurrent* if every subword of w occurs infinitely many times.

Equivalently, w is recurrent if every prefix of w occurs again as a subword. For instance, the Thue-Morse word is recurrent because every prefix occurs in some $t^n(0)$, and since 0 occurs infinitely many times in $\mathbf{t}$, $t^n(0)$ occurs infinitely many times in $t^n(\mathbf{t}) = \mathbf{t}$. On the other hand, the characteristic sequence of powers of 2 contains the subword 10001 exactly once, so it is clearly not recurrent. Finally, the *Cantor sequence*,

$$\mathbf{k} := ababbbababbbbbbbbbababbbaba \cdots,$$

defined as a fixed point of $a \mapsto aba, b \mapsto bbb$, is recurrent. However, some subwords of $\mathbf{k}$ (e.g., a or aba) do not occur in arbitrarily large subwords of $\mathbf{k}$. That is, the gap between consecutive occurrences of a is unbounded. This leads to another definition where the size of the gap is relevant.

Definition 1.24. An infinite recurrent word w is *uniformly recurrent* if for all $n \geq 0$ there exists a constant r_n such that every subword of length n occurs in any *window* (subword) of length r_n.

We define $r_w \colon \mathbb{N} \to \mathbb{N}$, the *recurrence function* of a uniformly recurrent word w as follows. Let $r_w(n)$ be the minimal window size such that every window of length $r_w(n)$ contains all subwords of length n, for all n.

If $r_w(n) \in O(n)$ then we say w is *linearly recurrent*. Given a linearly recurrent word w, we are naturally interested in the sup, lim sup, inf, lim inf, etc. of the ratio $r_w(n)/n$.

One application of the recurrence function is to enumerate subwords. Given a uniformly recurrent word z, we can enumerate the subwords of length n by constructing any window w (for simplicity, usually a prefix) of length $r_z(n)$ and then return all length n subwords in w. The recurrence function may be unnecessarily large for this application, so we introduce the appearance and condensation functions.

Definition 1.25. Let w be an infinite word. The *appearance function*, $A_w \colon \mathbb{N} \to \mathbb{N}$, is defined so that $A_w(n)$ is the length of the shortest prefix of w containing all subwords of length n in w. We define the *condensation function*, $C_w \colon \mathbb{N} \to \mathbb{N}$, so that $C_w(n)$ is the length of the shortest subword in w that contains all subwords of w of length n.

In Chapter 3, we will see how to compute these functions for k-automatic sequences, and identify the position of the shortest window for the condensation function. In Chapter 4, we show how to compute the minimal linear recurrence constant, and determine the behaviour of $\frac{A_w(n)}{n}$ and $\frac{C_w(n)}{n}$ in the limit as $n \to \infty$.

Chapter 2

Automatic Sequences

In this section, we cover the basic results on k-automatic sequences and $\mathcal{N}$-automatic sequences, so that we may refer to them in later chapters. We say a sequence $w \in \Gamma^\omega$ over a finite alphabet Γ is k-automatic (for $k \geq 2$) if there is a DFAO that computes the nth term of the sequence, given the base-k representation for n as input. Other numeration systems (i.e., other ways of representating $\mathbb{N}$ with finite strings) lead to a generalization called $\mathcal{N}$-automatic sequences, studied by Lecomte and Rigo [35] as S-automatic sequences. In particular, any morphic sequence is $\mathcal{N}$-automatic under an appropriate numeration system. There are also the k-regular and k-synchronized sequences, which generalize k-automatic sequences to unbounded sequences over $\mathbb{N}$. In this chapter we discuss numeration systems, then general $\mathcal{N}$-automatic sequences. Then we cover theorems for the special case of k-automatic sequence, and finally k-synchronized and k-regular sequences.

2.1 Numeration Systems

A numeration system is method for representing elements of $\mathbb{N}$ as finite words.

Definition 2.1. A *numeration system* is a triple $(\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ consisting of

- a finite alphabet, Σ,
- a language $L \subseteq \Sigma^*$, and
- a surjective function $\langle\cdot\rangle_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$.

Words in L are called *representations*, and we say $w \in L$ *is a representation for an integer* $n \geq 0$ if $\langle w\rangle_{\mathcal{N}} = n$.

Everyone is familiar with the base-10 numeration system, but there are many others.

- Base-r representation for $r \geq 2$. The numeration system is $\mathcal{N}_r = (\Sigma_r, \Sigma_r^*, \langle\cdot\rangle_r)$, where $\Sigma_r := \{0, 1, \ldots, r-1\}$ and

$$\langle a_0 a_1 \cdots a_{n-1} \rangle_r := \sum_{i=0}^{n-1} a_i r^i.$$

  For example, the word $329 \in \Sigma_{10}^*$ is a base-10 representation for the number $923 \in \mathbb{N}$. Note that the digits are reversed because this is the *least significant digit first* base-r representation. There is also the (more familiar) *most significant digit first* base-r representation. We discuss the most significant digit/least significant digit issue further in the next section.
- Fibonacci (Zeckendorf) representation. Let $\mathcal{F} = (\{0,1\}, L_\mathcal{F}, v_\mathcal{F})$ where $L_\mathcal{F} \subseteq \{0,1\}^*$ is the set of words that do not contain 11, and

$$v_F(d_0 d_1 \cdots d_n) = \sum_{i=0}^{n} d_i F_{i+2}$$

  where $(F_i)_{i=0}^\infty$ is the Fibonacci sequence,

$$F_n = \begin{cases} 0, & \text{if } n = 0; \\ 1, & \text{if } n = 1; \\ F_{n-1} + F_{n-2}, & \text{if } n \geq 2. \end{cases}$$

- Bijective base-r representation for $r \geq 1$. Define the numeration system $\mathcal{B}_r = (\{1, \ldots, r\}, \{1, \ldots, r\}^*, v_r)$ where

$$v_r(a_{n-1} a_{n-2} \cdots a_1 a_0) = \sum_{i=0}^{n-1} a_i r^i,$$

  as before. It turns out that $v_r \colon \{1, \ldots, r\}^* \to \mathbb{N}$ is a bijection. Note that when $r = 1$, bijective base-r representation is a unary representation.

- Morphic numeration system. Given a morphism and its infinite fixed point, we construct a numeration system based on prefixes of the fixed point. Since this system is somewhat complicated to construct, we present it separately in Section 2.2.2.

- Roman numerals. Let $\Sigma = \{\mathtt{I}, \mathtt{V}, \mathtt{X}, \mathtt{L}, \mathtt{C}, \mathtt{D}, \mathtt{M}\}$ and define the numeration system $\mathcal{R} = (\Sigma, L, v)$.
  Define the maps $h_{\mathtt{IVX}}, h_{\mathtt{XLC}}, h_{\mathtt{CDM}} \colon \Sigma_{10} \to \Sigma^*$ where

$$\begin{aligned} h_{\mathtt{IVX}}(0) &= \varepsilon & h_{\mathtt{IVX}}(5) &= \mathtt{V} \\ h_{\mathtt{IVX}}(1) &= \mathtt{I} & h_{\mathtt{IVX}}(6) &= \mathtt{VI} \\ h_{\mathtt{IVX}}(2) &= \mathtt{II} & h_{\mathtt{IVX}}(7) &= \mathtt{VII} \\ h_{\mathtt{IVX}}(3) &= \mathtt{III} & h_{\mathtt{IVX}}(8) &= \mathtt{VIII} \\ h_{\mathtt{IVX}}(4) &= \mathtt{IV} & h_{\mathtt{IVX}}(9) &= \mathtt{IX}, \end{aligned}$$

  with $h_{\mathtt{XL}}$ and $h_{\mathtt{CD}}$ defined similarly. Then we define

$$L = \{\mathtt{M}^i h_{\mathtt{CDM}}(a_2) h_{\mathtt{XLC}}(a_1) h_{\mathtt{IVX}}(a_0) : a_0, a_1, a_2 \in \Sigma_{10}, i \geq 0\}$$

  and

$$v(\mathtt{M}^i h_{\mathtt{CDM}}(a_2) h_{\mathtt{XLC}}(a_1) h_{\mathtt{IVX}}(a_0)) = 1000i + 100a_2 + 10a_1 + a_0.$$

- Base-$(-r)$ representation for $r \geq 2$. We define $\mathcal{N}_{-r} = (\Sigma_r, \Sigma_r^*, \langle\cdot\rangle_{-r})$ where

$$\langle a_0 \cdots a_{n-1} \rangle_{-r} := \sum_{i=0}^{n-1} (-r)^i a_i.$$

  Note that $\langle\cdot\rangle_{-r}$ maps some representations to negative integers, so base-$(-r)$ is technically not a numeration system. One can revise our definition of numeration systems to include representations for elements $\mathbb{Z}$, or $\mathbb{Z}[i]$, and so on, but we will not do this formally. It is known that every element of $\mathbb{Z}$ has a representation in base-$(-r)$ for $r \geq 2$.

2.2 Properties of Numeration Systems

We have defined numeration systems so that almost any mapping between words and $\mathbb{N}$ qualifies as a numeration system. Hence, the concept of a numeration system is not very useful unless we require a few additional properties.

- First, we are only interested in numeration systems that represent *all* numbers in $\mathbb{N}$, for obvious reasons. That is, we require $\langle\cdot\rangle_{\mathcal{N}}$ to be surjective. In the literature, a numeration system that has a representation for all numbers is sometimes called *complete*.

- We say a numeration system is *ambiguous* if there is more than one representative for some integer. That is, if there exist $x, y \in L$ such that $\langle x \rangle_{\mathcal{N}} = \langle y \rangle_{\mathcal{N}}$ but $x \neq y$. A numeration system is *unambiguous* if it is not ambiguous. When a numeration system is both unambiguous and complete, we say it is *perfect*.
  In a perfect numeration system, $\langle \cdot \rangle_{\mathcal{N}}$ is bijective, and it is useful to define the inverse, $(\cdot)_{\mathcal{N}}$. We will assume the existence of this function when $\mathcal{N}$ is a perfect numeration system.

- Let $\mathcal{N} = (\Sigma, L, \langle \cdot \rangle_{\mathcal{N}})$ be a numeration system. If there exists a symbol $0 \in \Sigma$ such that L is of the form $L'0^*$, and $(\Sigma, L', \langle \cdot \rangle_{\mathcal{N}}|_{L'})$ is a perfect numeration system, then we say $\mathcal{N}$ is a *typical least significant digit first* numeration system or *typical LSD* numeration system. Similarly, if L is instead of the form $0^*L'$ then we say $\mathcal{N}$ is a *typical most significant digit first* numeration system or *typical MSD* numeration system.

- Let $\mathcal{N} = (\Sigma, L, \langle \cdot \rangle_{\mathcal{N}})$ be a numeration system. We say $\mathcal{N}$ is *automatic* if L is a regular language. This will be important later.

We say a numeration system $\mathcal{N}$ is *ideal* if it is automatic and typical (either typical MSD or typical LSD). We will almost exclusively use ideal numeration systems. The advantage of an ideal numeration system is that there is essentially just one canonical representation for each $n \in \mathbb{N}$, with optional padding. In an ideal numeration system, we let $(n)_{\mathcal{N}}$ denote the shortest representation for a number $n \in \mathbb{N}$.

Recall that a numeration system is automatic if the language of representations is regular; in other words, if we can recognize representations with an automaton. We say an operation (e.g., comparison or addition) is *automatic* if there is an automaton that computes that operation in some sense. The precise definition varies depending on the operation.

- Suppose $\mathcal{N}$ is an ideal numeration system. It follows immediately that equality comparison is automatic. That is, the language

  $$L_= := \{(x, y) \in L \times L : \langle x \rangle_{\mathcal{N}} = \langle y \rangle_{\mathcal{N}}\}$$

  is regular. Recall that an automaton with multiple inputs (in this case, x and y) uses padding to ensure the inputs are the same length, and since $\mathcal{N}$ is an ideal numeration system, we use 0 as the padding symbol. In an ideal numeration system, a number has at most one representation of a given length, so the two inputs to our automaton represent the same number if and only if they are identical.

- We say *comparison is automatic* in $\mathcal{N}$ if the following language is regular.

$$L_< := \{(x, y) \in L \times L : \langle x \rangle_{\mathcal{N}} < \langle y \rangle_{\mathcal{N}}\}.$$

  Note that comparison is automatic for many numeration systems. In fact, in many systems, comparison of representations is the same as lexicographic comparison, starting from the most significant digit. For instance, 737 is less than 1002 (in base 10) and 0737 is lexicographically less than 1002, using the usual ordering on digits. We say an ideal numeration system $\mathcal{N}$ has *lexicographic comparison* if $x \leq y \Leftrightarrow \langle x \rangle_{\mathcal{N}} \leq \langle y \rangle_{\mathcal{N}}$ for all representations $x, y \in L$ such that $|x| = |y|$. We claim without proof that base-k representation, Fibonacci representation and morphic numeration systems all have lexicographic comparison.

- We say *addition is automatic* in $\mathcal{N}$ if

$$L_+ := \{(x, y, z) \in L \times L \times L : \langle x \rangle_{\mathcal{N}} + \langle y \rangle_{\mathcal{N}} = \langle z \rangle_{\mathcal{N}}\}$$

  is a regular language. There is a remarkably simple automaton for addition in base-k representation (see Figure 2.1), and addition is automatic in the Fibonacci representation (see [27]). On the other hand, addition is not automatic in unary numeration systems and some morphic numeration systems.
  Suppose that L_+ is regular with a unary numeration system. The unary representation of $2n$ (i.e., $(n)_1$) grows arbitrarily long relative to the unary representation of n. For sufficiently large n, we can apply the pumping lemma $((n)_1, (n)_1, (2n)_1)$ in such a way that we only change the last value, $(2n)_1$. Since there is only one representation (ignoring padding) of $n + n = 2n$, the automaton accepts triples (x, y, z) such that $\langle x \rangle_1 + \langle y \rangle_1 \neq \langle z \rangle_1$.
  In later chapters, we will need addition to state many interesting questions about sequences, and we will need automatic addition in the corresponding numeration system to mechanically resolve those questions. Note that sometimes we only need addition by a constant (e.g., successor of a number), but this is much weaker than full addition. We argue in Chapter 3 that addition by a constant is automatic if comparison is automatic.

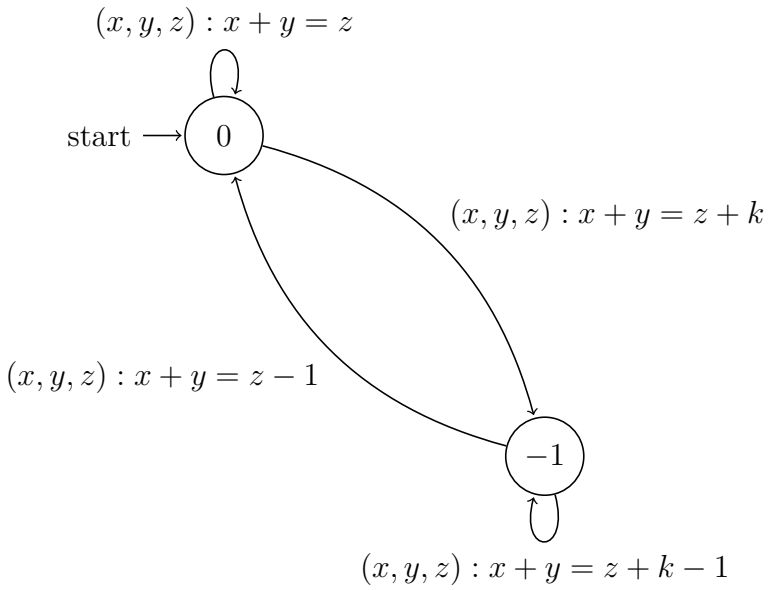


Figure 2.1: Addition in base k with LSD first.

2.2.1 ω-Numeration Systems

It is possible to use infinite words, instead of finite words, to represent numbers. An ω-*numeration system* is a triple $(\Sigma, L_\omega, \langle\cdot\rangle_{\mathcal{N}_\omega})$ where Σ is a finite alphabet, $L_\omega \subseteq \Sigma^\omega$ is an ω-language of infinite words, $\langle\cdot\rangle_{\mathcal{N}_\omega} : L_\omega \to \mathbb{N}$ is a map from representations to numbers. We will not get into a discussion of general ω-numeration systems; we are only interested ω-numeration systems corresponding to ideal numeration systems.

Definition 2.2. Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ be a LSD ideal numeration system with padding symbol $0 \in \Sigma$. We define an associated ω-*numeration system* $\mathcal{N}_\omega$, where the set of representations is

$$L_\omega := \{x0^\omega : x \in L\} \subseteq \Sigma^\omega$$

and the number $n \in \mathbb{N}$ is associated with the representation $(n)_{\mathcal{N}} 0^\omega \in L_\omega$. We define bijections $\langle\cdot\rangle_{\mathcal{N}_\omega} : L_\omega \to \mathbb{N}$ and $(\cdot)_{\mathcal{N}_\omega} : \mathbb{N} \to L_\omega$ between L_ω and $\mathbb{N}$.

Theorem 2.3. *Let $\mathcal{N}$ be a LSD ideal numeration system. If an operation is automatic in $\mathcal{N}$ then the same operation is automatic in $\mathcal{N}_\omega$. That is, L_ω is ω-regular and so is*

$$L_= := \{(x, y) \in L_\omega^2 : \langle x\rangle_{\mathcal{N}_\omega} = \langle y\rangle_{\mathcal{N}_\omega}\}.$$

Additionally,

- *If comparison is automatic in $\mathcal{N}$ then*

$$L_< := \{(x, y) \in L_\omega^2 : \langle x \rangle_{\mathcal{N}_\omega} \leq \langle y \rangle_{\mathcal{N}_\omega}\}$$

  is ω-regular.
- *If addition is automatic in $\mathcal{N}$ then*

$$L_+ := \{(x, y, z) \in L_\omega^3 : \langle x \rangle_{\mathcal{N}_\omega} + \langle y \rangle_{\mathcal{N}_\omega} = \langle z \rangle_{\mathcal{N}_\omega}\}$$

  is ω-regular.

Proof. Recall that if $A \subseteq \Sigma^*$ is a regular language and $B \subseteq \Sigma^\omega$ is an ω-regular language, then $AB = \{ab : a \in A, b \in B\} \subseteq \Sigma^\omega$ is an ω-regular language by definition. The claims follow from the fact that L_ω, $L_=$, $L_<$ and L_+ can be constructed by appending the ω-regular language $\{0^\omega\} \subseteq \Sigma^\omega$ to the corresponding (regular) languages over finite words. □

2.2.2 Morphic Numeration Systems

Given a morphic word, one can define a corresponding morphic numeration system. This numeration system will be useful in Chapter 3, to show that all morphic words are $\mathcal{N}$-automatic (previously shown by Rigo [45]), and later in Chapter 5, where the numeration system gives us a way to describe subwords. This numeration system can be traced to Rigo.

Definition 2.4. Suppose $\varphi\colon \Gamma^* \to \Sigma^*$ is a morphism, and we have words $u \in \Sigma^*$, $v \in \Gamma^*$ such that $u = \varphi(v)$.

For x a finite prefix of u, define $x \operatorname{div} \varphi \in \Gamma^*$ to be the longest prefix of v such that $\varphi(x \operatorname{div} \varphi)$ is a prefix of x. Then define $x \bmod \varphi \in \Sigma^*$ so that

$$x = \varphi(x \operatorname{div} \varphi)(x \bmod \varphi).$$

Naturally, we call x the *dividend*, φ the *divisor*, $x \operatorname{div} \varphi$ the *quotient* and $x \bmod \varphi$ the *remainder*.

Note that the definition of $x \operatorname{div} \varphi$ depends on u and v, which are not part of the notation. For instance, if $\varphi\colon \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \to \{\mathtt{0}, \mathtt{1}\}^*$ is such that

$$\begin{aligned}\varphi(\mathtt{a}) &= \mathtt{01}\\ \varphi(\mathtt{b}) &= \mathtt{0}\\ \varphi(\mathtt{c}) &= \mathtt{1}.\end{aligned}$$

Then $x = \texttt{01}$ is a prefix of $u_1 = \varphi(\texttt{a}\cdots)$ and $u_2 = \varphi(\texttt{bc}\cdots)$. Depending on whether we interpret x as a prefix of u_1 or as a prefix of u_2, the quotient is either $\texttt{a}$ or $\texttt{bc}$. Typically, we consider quotients and remainders in the special case that $u = \varphi(v) = v$ is a morphic fixed point. There is still some ambiguity, since φ may have multiple fixed points (especially if φ is erasing), but u and v are usually clear from context.

The remainder in the division x div φ has a special form which depends on φ.

Proposition 2.5. *Suppose $\varphi\colon \Sigma^* \to \Sigma^*$ is a non-erasing morphism and $w \in \Sigma^\omega$ is a fixed point of φ. For any prefix $x \in \Sigma^*$ of $w := \varphi^\omega(c)$, the remainder, $x \bmod \varphi$, is a proper prefix of $\varphi(a)$ where $a \in \Sigma$ is a symbol such that $(x \operatorname{div} \varphi)a$ is a prefix of w, contradicting the maximality of $x \operatorname{div} \varphi$*

Proof. Let $q = x \operatorname{div} \varphi$ and $r = x \bmod \varphi$. Since q is a prefix of w, we can write $w = qaz$ for some $a \in \Sigma$ and $z \in \Sigma^\omega$. Then $\varphi(qa)$ is a prefix of $w = \varphi(w)$, but is not a prefix of x, since we took q as long as possible such that $\varphi(q)$ is a prefix of x. Therefore $x = \varphi(q)r$ is a prefix of $\varphi(qa)$, and hence r is a prefix of $\varphi(a)$. Furthermore, $r \neq \varphi(a)$ since then $\varphi(qa)$ would be a prefix of x. □

Definition 2.6. Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. We define a language $\Delta_\varphi \subseteq \Sigma^*$ where

$$\Delta_\varphi := \{x \in \Sigma^* : x \text{ is a prefix of } \varphi(c) \text{ for } c \in \Sigma\}.$$

Observe that Δ_φ is a finite set. Given a prefix x of w, we define a sequence $\mathsf{D}(x) \in \Delta_\varphi^\omega$ where

$$\mathsf{D}(x) = (x \bmod \varphi)\mathsf{D}(x \operatorname{div} \varphi)$$

Note that $\mathsf{D}(x)$ is a word of words, since elements of Δ_φ are finite words. This is particularly confusing because the empty word, ε, is always a member of Δ_φ. For example, in the following proposition, we argue that most of the symbols in D are ε.

Proposition 2.7. *Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w = \varphi^\omega(c) \in \Sigma^\omega$. Let Δ_φ and D be as defined earlier. Then all but the first n symbols in $\mathsf{D}(w[0..n-1])$ are ε, for all $n \geq 0$.*

Proof. Observe that $|\varphi(c)| > 1$ since $\varphi^\omega(c)$ is an infinite fixed point. Furthermore, $|\varphi(a)| \geq 1$ for all $a \in \Sigma$ since φ is non-erasing. Therefore $|\varphi(w[0..n-1])| \geq n$ for all n, with equality if and only if $n = 0$.

Now we prove that $\mathsf{D}(w[0..n-1])[m] = \varepsilon \in \Delta_\varphi$ for all $m \geq n$ by induction on n. It is trivial to see that $\mathsf{D}(\varepsilon) = \varepsilon^\omega$, so the induction hypothesis holds when $n = 0$. When $n > 0$, we see that $\mathsf{D}(w[0..n-1])[m] = \mathsf{D}(w[0..n-1] \operatorname{div} \varphi)[m-1]$, and since $w[0..n-1] \operatorname{div} \varphi$ has length at most $n-1$, the induction hypothesis says that $\mathsf{D}(w[0..n-1] \operatorname{div} \varphi)[m-1] = \varepsilon$ for all $m - 1 \geq n - 1$, completing the induction. □

We define a numeration system $\mathcal{N}_\varphi$ based on $\mathsf{D}(w[0..n-1])$.

Definition 2.8. Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. Let D and Δ_φ be as defined earlier. Let $\ell\colon \Delta_\varphi^* \to \Sigma_k^*$ be a coding such that $\ell(a) = |a|$ for all $a \in \Delta_\varphi$. That is, ℓ maps each word in Δ_φ^* (recall that the symbols are words) to a digit by taking the length. Then we define the *morphic numeration system of* w, $\mathcal{N}_\varphi = (\Sigma_k, L, \langle\cdot\rangle_\varphi)$, where

- $\Sigma_k = \{0, 1, \cdots, k-1\}$ where $k = \max_{a\in\Sigma} |\varphi(a)| = 1 + \max_{a\in\Delta_\varphi} \ell(a)$,
- $L = \{x \in \Sigma_k^* : x0^\omega = \ell(\mathsf{D}(w[0..n-1])) \text{ for some } n \geq 0\}$, and
- $\langle x\rangle_\varphi = n$ for $x \in L$ if $x0^\omega = \ell(\mathsf{D}(w[0..n-1]))$.

Hypothetically, $\langle\cdot\rangle_\varphi$ could be ill-defined, since $x \in L$ may be a prefix of more than one word of the form $\ell(\mathsf{D}(w[0..n-1]))$. We address this by defining a total order $\prec$ on the set L, where

$$x \prec y \iff x[i..\infty] = y[i..\infty] \text{ and } x[i] < y[i] \text{ for some } i,$$

for all $x, y \in L$. Then the following theorem shows that $\langle\cdot\rangle_\varphi$ is well-defined.

Theorem 2.9. *Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. Let $\mathcal{N}_\varphi = (\Sigma_k, L, \langle\cdot\rangle_\varphi)$ be the morphic numeration system of w. Suppose $m, n \geq 0$ are natural numbers. Then $m < n$ if and only if $\ell(\mathsf{D}(w[0..m-1])) \prec \ell(\mathsf{D}(w[0..n-1]))$. It follows that $\ell(\mathsf{D}(w[0..m-1])) = \ell(\mathsf{D}(w[0..n-1]))$ if and only if $m = n$.*

Proof. It suffices to prove that $\ell(\mathsf{D}(w[0..n-1])) \prec \ell(\mathsf{D}(w[0..n]))$, for all n. We prove this by induction on n. When $n = 0$, it is trivial since $\ell(\mathsf{D}(w[0])) = 10^\omega$ and $\mathsf{D}(\varepsilon) = 0^\omega$. When $n > 0$, let $w[0..m-1] := w[0..n-1] \operatorname{div} \varphi$ and observe that $w[0..n] \operatorname{div} \varphi$ is either $w[0..m-1]$ or $w[0..m]$, since φ is non-erasing. If $w[0..m-1] = w[0..n] \operatorname{div} \varphi$ then $w[0..n] \bmod \varphi$ must be longer than $w[0..n-1] \bmod \varphi$ so the first digit of $\ell(\mathsf{D}(w[0..n]))$ is larger than the least digit of $\ell(\mathsf{D}(w[0..n-1]))$, and all other digits are the same. If $w[0..m] = w[0..n] \operatorname{div} \varphi$ then by the induction hypothesis, $\ell(\mathsf{D}(w[0..m-1])) \prec \ell(\mathsf{D}(w[0..m]))$, and the result follows. □

As a corollary, we see that $\mathcal{N}_\varphi$ is an LSD typical numeration system with lexicographic comparison, since $x0^\omega \prec y0^\omega$ is the same as lexicographic comparison of x and y, as long as x and y are the same length.

Corollary 2.10. *Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. Let $\mathcal{N}_\varphi = (\Sigma_k, L, \langle\cdot\rangle_\varphi)$ be the morphic numeration system of w. Then $\mathcal{N}_\varphi$ is an LSD typical numeration system and comparison is lexicographic in $\mathcal{N}_\varphi$.*

Proof. Let $n \geq 0$ be an arbitrary natural number. By Proposition 2.7, $\ell(\mathsf{D}(w[0..n-1])) = x0^\omega$ for some $x \in L$. Let us assume x is as short as possible. Then $x0^i$ is a $\mathcal{N}_\varphi$-representation of n for all $i \geq 0$. Conversely, these are clearly the only representations of n, since any representation is a prefix of $\ell(\mathsf{D}(w[0..n-1])) = x0^\omega$, and x is the shortest prefix in L. □

In fact, $\mathcal{N}_\varphi$ is an ideal numeration system. A consequence of the following theorem is that the language of morphic decompositions is regular, and therefore the set of representations in $\mathcal{N}_\varphi$ is regular.

Theorem 2.11. *Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. Then there exists a DFAO $M\colon \Delta_\varphi^* \to \Sigma \cup \{\varnothing\}$ such that*

$$M(x) = \begin{cases} w[n], & \text{if } x\varepsilon^\omega = \mathsf{D}(w[0..n-1]), \\ \varnothing, & \text{otherwise,} \end{cases}$$

for $x \in \Delta_\varphi^*$.

Proof. We construct an automaton $T\colon \Delta_\varphi^* \to \Sigma \cup \{\varnothing\}$ such that $T(x) = M(x^R)$, and rely on the closure properties of DFAOs to prove that M exists. We let $T = (\Delta_\varphi, \Sigma \cup \{\varnothing\}, \delta, q_0, \Sigma \cup \{\varnothing\}, \gamma)$ so that

- the set of states is $\Sigma \cup \{\varnothing\}$,
- the initial state is $q_0 = w[0]$,
- the output associated with a state q is $\gamma(q) := q$,
- there is a transition from $a \in \Sigma$ to $b \in \Sigma$ on input $v \in \Delta_\varphi$ if vb is a prefix of $\varphi(a)$. All other transitions go to state $\varnothing$.

Suppose that $\mathsf{D}(w[0..n-1]) = d_0 d_1 \cdots d_{j-1}\varepsilon^\omega$ for $d_0 d_1 \cdots d_{j-1} \in \Delta_\varphi^*$. We will show by induction on j that $T(d_{j-1} \cdots d_1 d_0) = w[n]$. Clearly it is true when $j = 0$, since then $\mathsf{D}(w[0..n-1]) = \varepsilon^\omega$ so $n = 0$, and we know that $T(\varepsilon) = q_0 = w[0]$. If $j > 0$ then observe that

$$\begin{aligned} w[0..n-1] &= \varphi(w[0..n-1] \operatorname{div} \varphi)(w[0..n-1] \bmod \varphi) \\ &= \varphi(w[0..m-1])d_0 \end{aligned}$$

where $w[0..m-1] = w[0..n-1] \operatorname{div} \varphi$. Then $d_1 \cdots d_{j-1}\varepsilon^\omega = \mathsf{D}(w[0..m-1])$, so by the induction hypothesis, $T(d_{j-1} \cdots d_1) = w[m]$. By Proposition 2.5, $d_0 w[n]$ is a prefix of $\varphi(w[m])$, so there is a transition from $w[m]$ to $w[n]$ on d_0. It follows that $T(d_{j-1} \cdots d_1 d_0) = w[n]$, completing the induction.

A similar induction shows that if $T(d_{j-1} \cdots d_0) \neq \varnothing$ then $d_0 d_1 \cdots d_{j-1}\varepsilon^\omega$ is of the form $\mathsf{D}(w[0..n-1])$ for some $n \in \mathbb{N}$. We leave this as an exercise to the reader. □

Corollary 2.12. *Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a non-erasing morphism with fixed point $w \in \Sigma^\omega$. Let $\mathcal{N}_\varphi = (\Sigma_k, L, \langle\cdot\rangle_\varphi)$ be the morphic numeration system of w. Then the language L is regular (so $\mathcal{N}_\varphi$ is automatic) and there is a DFAO $M\colon \Sigma_k^* \to \Sigma$ such that $M(x) = w[\langle x\rangle_\varphi]$ for all $x \in L$.*

Proof. Use the automaton M from the previous theorem, except replace the transition labels (words in Δ_φ) with their lengths. The automaton is still deterministic because for each state $q \in \Sigma$ in the automaton, the outgoing edges correspond to prefixes of $\varphi(q)$, and there is at most one prefix of any given length. □

The fact that $x \mapsto w[\langle x\rangle_\varphi]$ is an automatic function leads us into the next section, where we study sequences with the same property in other numeration systems.

2.3 $\mathcal{N}$-Automatic Sequences

We begin with k-automatic sequences, which are a special case of $\mathcal{N}$-automatic sequences.

Definition 2.13. Let $k \geq 2$ be an integer. We say a sequence $w \in \Gamma^\omega$ is *k-automatic* if the function $f\colon \Sigma_k^* \to \Gamma$ such that

$$f(x) = w[\langle x\rangle_k]$$

is automatic.

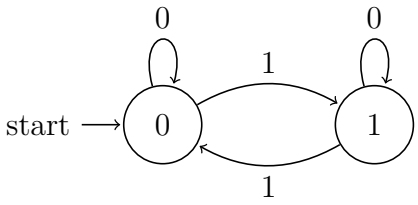


Figure 2.2: Automaton for the Thue-Morse word

The Thue-Morse word is the canonical example of a a 2-automatic sequence, and is computed by the DFAO in Figure 2.2. The period-doubling word, Rudin-Shapiro word, and Baum-Sweet word from the previous chapter are also examples of 2-automatic sequences, and the Mephisto waltz word is 3-automatic. All of these examples follow from Cobham's theorem, below, and the morphic definitions in the previous chapter.

Theorem 2.14 (Cobham)**.** *Let $k \geq 2$ be an integer. A sequence $w \in \Gamma^\omega$ is k-automatic if and only if it is of the form $h(\varphi^\omega(c))$ for some*

- *finite alphabet Δ,*
- *coding $h\colon \Delta^* \to \Gamma^*$,*
- *uniform morphism $\varphi\colon \Gamma^* \to \Gamma^*$, and*
- *symbol $c \in \Delta$.*

Proof. See [18] or [6, Theorem 6.3.2, p. 175]. Alternatively, one direction follows from the fact that the morphic numeration system associated with a k-uniform morphism is equivalent to the base-k numeration system, $\mathcal{N}_k$. Then Corollary 2.12 provides the desired automaton. □

We mention another remarkable theorem about k-automatic sequences, also due to Cobham, before generalizing to $\mathcal{N}$-automatic sequences.

Theorem 2.15. *Let $k, \ell \geq 2$ be integers. We say k and ℓ are* multiplicatively dependent *if there exist positive integers a, b such that $k^a = \ell^b$.*

Suppose that k, ℓ are multiplicatively independent (i.e., not multiplicatively dependent). Then a sequence $w \in \Gamma^\omega$ both is k-automatic and ℓ-automatic if and only if w is ultimately periodic.

Proof. See [17] or [6, Theorem 11.2.2, p. 350]. □

Definition 2.16. Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ be a numeration system. A word $w \in \Gamma^*$ is $\mathcal{N}$-automatic if the function $f\colon \Sigma^* \to \Gamma$ given by $f(x) = w[\langle x\rangle_{\mathcal{N}}]$ is automatic.

Here are some examples of $\mathcal{N}$-automatic sequences.

- Any morphic word $w = h(\varphi^\omega(c))$ is $\mathcal{N}$-automatic, where $\mathcal{N}$ is the morphic numeration system corresponding to φ and $\varphi^\omega(c)$. The Fibonacci word is a nice example, where

we use Zeckendorf representation as our numeration system. In Chapter 5 we use the $\mathcal{N}$-automatic structure of the fixed point of

$$\begin{aligned} 0 &\mapsto 03 \\ 1 &\mapsto 43 \\ 3 &\mapsto 1 \\ 4 &\mapsto 01, \end{aligned}$$

to show that it avoids a certain pattern.

- Define the *Champernowne word* (or *Barbier word* [6, Exercise 26, p. 114]),

$$\mathbf{c} = 0123456789101112131415 16 \cdots,$$

obtained by concatenating $0, 1, 2, \ldots$ written in base 10. We define a perfect numeration system $\mathcal{N} = (\{0, 1, \ldots, 9, \dot{0}, \dot{1}, \ldots, \dot{9}\}, L, \langle\cdot\rangle_{\mathcal{N}})$ where L is the language of all canonical base-10 representations (i.e., $0, 1, \cdots, 10, 11, \cdots$) where one digit has been marked with a dot. For instance, $\dot{0}$, $1\dot{5}$, or $1\dot{0}3$, but not 12, $\dot{1}\dot{2}$, or $\dot{0}13$. Then we order the words in L as follows.

$$\dot{0} < \dot{1} < \cdots < \dot{9} < \dot{1}0 < 1\dot{0} < \dot{1}1 < 1\dot{1} < \dot{1}2 < 1\dot{2} < \cdots$$

That is, we compare to words by padding them to the same length with $\dot{0}$ on the left, and then comparing lexicographically according to the ordering

$$\dot{0} < 0 < \dot{1} < 1 < \cdots < \dot{9} < 9$$

on the alphabet. We let the nth word in this lexicographic order represent $n - 1$. It follows that $\mathbf{c}$ is $\mathcal{N}$-automatic, since we can construct a DFAO that returns the marked digit. Note that $\mathbf{c}$ is not a morphic word because it has subword complexity 10^n.

- The characteristic sequence of $\{n2^n\}_{n=0}^{\infty}$ is $\mathcal{N}$-automatic, where $\mathcal{N}$ is more or less any numeration system with exactly $n2^n$ representations of length at most n. Hence, it has $n2^n - (n-1)2^{n-1} = 2^n + (n-1)2^{n-1}$ representations of length exactly n. For instance, a language that includes all nonempty binary words, $\{0, 1\}^+$, and binary words where one non-leading symbol has been replaced with a placeholder symbol, say, X. That is,

$$L = \{0, 1, 00, 01, 0X, 10, 11, 1X, 000, 001, 00X, 010, 011, 01X, 0X0, 0X1, \ldots\}$$

Then 0^{n+1} is the representation for $n2^n$, so the characteristic sequence of $\{n2^n\}_{n=0}^{\infty}$ is $\mathcal{N}$-automatic.

We will also encounter multidimensional k-automatic and $\mathcal{N}$-automatic sequences.

2.4 k-Regular and k-Synchronized Functions

A limitation of k-automatic sequences is that the domain must be a finite set, bounded in size by the number of states in the corresponding DFAO. We will discuss two generalizations of k-automatic sequences to unbounded subsets of $\mathbb{N}$, or in some cases $\mathbb{Z}^d$.

2.4.1 k-Regular Functions

First we have k-regular sequences, or more generally, k-regular functions. The concept of k-regular sequences is due to Allouche and Shallit [5, 6].

Definition 2.17. Let $f\colon \mathbb{N}^n \to Z^m$ be a function, and let $k \geq 2$ be an integer. The *k-kernel* of f is the set

$$S := \{(i_1, \ldots, i_n) \mapsto f(k^d i_1 + a_1, \ldots, k^d i_n + a_n) : d \in \mathbb{N}, 0 \leq a_i < k^d \text{ for all } i\}.$$

We say f is *k-regular* if the $\mathbb{Z}$-module generated by the k-kernel is finitely generated.

For example, the function $n \mapsto n$ is a k-regular sequence because the k-kernel,

$$\{n \mapsto kn + a : d \in \mathbb{N}, 0 \leq a < k^d\},$$

generates the $\mathbb{Z}$-module $\{n \mapsto (an + b) : a, b \in \mathbb{Z}\}$, which is finitely generated by the elements $n \mapsto n$ and $n \mapsto 1$. Every k-automatic sequence over a subset of $\mathbb{Z}$ is k-regular, by the following theorem.

Theorem 2.18. *Let $f\colon \mathbb{N}^n \to \mathbb{Z}^m$ be a k-regular function. Then f is k-automatic if and only if f is bounded.*

Proof. See [6, Theorem 16.1.5, p. 441]. □

The Thue-Morse sequence, for example, gives us another example of a 2-regular function, $n \mapsto t[n]$.

Theorem 2.19. *The set of k-regular functions of the form $\mathbb{N}^n \to \mathbb{Z}^m$ is closed under element-wise addition, scalar multiplication and element-wise multiplication. The constant 0 sequence and constant 1 sequence serve as the additive identity and multiplicative identity respectively. Therefore the k-regular functions form a ring.*

Proof. See [6, Theorem 16.2.1, p. 441]. □

Corollary 2.20. *Let $f\colon \mathbb{N} \to \mathbb{Z}^m$ be a function, and let $g\colon \mathbb{N} \to \mathbb{Z}^m$ be the first difference function, $g(n) = f(n+1) - f(n)$. Then f is k-regular if and only if g is k-regular.*

Proof. If the k-kernel for $f(n)$ is finitely generated, then so is the k-kernel of $f(n+1)$, and together they generate the k-kernel of $g(n)$. Conversely, if the k-kernel of $g(n)$ is finitely generated then □

It follows that any integer polynomial is an example of a k-regular function for all $k \geq 2$.

Another interesting example arises in the analysis of Karatsuba's algorithm for polynomial multiplication. The unique solution to *Karatsuba's recurrence*, shown below, counts the exact number of ring operations to compute the product of two degree $n-1$ polynomials using Karatsuba's algorithm, assuming we revert to the naive algorithm when it is more efficient.

$$T(n) = \begin{cases} 0, & \text{if } n = 0; \\ 2n^2 + 2n - 1, & \text{if } n = 1, 2, 3, 4, 5, 7; \\ T(\lfloor n/2 \rfloor) + 2T(\lceil n/2 \rceil) + 2\lfloor n/2 \rfloor + 3n - 4, & \text{otherwise.} \end{cases}$$

It turns out that the solution to this recurrence is 2-regular.

2.4.2 k-Synchronized Sequences

The second generalization of k-automatic is the class of k-synchronized functions. Note that k-synchronized sequences are due to Carpi and Maggi in [13].

Definition 2.21. We say a function $f\colon \mathbb{N} \to \mathbb{N}$ is *k-synchronized* if the *graph* of the function,

$$\{(n, f(n)) : n \in \mathbb{N}\} \subseteq \mathbb{N} \times \mathbb{N},$$

is k-automatic.

Similarly, a function $f\colon \mathbb{N}^n \to N^m$ is *k-synchronized* if the graph

$$\{(i_1, \ldots, i_n, j_1, \ldots, j_m) \in \mathbb{N}^{n+m} : f(i_1, \ldots, i_n) = (j_1, \ldots, j_m)\}$$

is k-synchronized.

For example, the functions $n \mapsto n+1$, $n \mapsto 2n$, $n \mapsto 3$ are all k-synchronized for every $k \geq 2$, because we can construct automata that accept their graphs. The function $(i,j) \mapsto i+j$ (i.e., the addition table) is an example of a 2-ary function that is k-synchronized. Finally, the function $(i,j) \mapsto i \oplus j$ is 2-synchronized, where $\oplus$ denotes the exclusive OR of the binary representations of i and j (also known as nimber addition, see [19]).

The first main theorem about k-synchronized functions relates them to k-regular functions.

Theorem 2.22. *Every k-synchronized function is also k-regular, for $k \geq 2$.*

As a corollary, any bounded k-synchronized function is k-automatic, by Theorem 2.18. On the other hand, every k-synchronized function is in $O(n)$.

Theorem 2.23. *Let $k \geq 2$ be an integer and let $f\colon \mathbb{N} \to \mathbb{N}$ be a k-synchronized function. Then $f(n)$ is in $O(n)$.*

Proof. Consider an automaton for the graph,

$$\{(n, f(n)) : n \in \mathbb{N}\} \subseteq \mathbb{N}^2.$$

Let s be the number of states in the automaton.

Suppose that $f(n)$ is not in $O(n)$, so $\frac{f(n)}{n}$ grows arbitrarily large. For some n, $f(n) \geq k^{s+1}n$ and hence n has at least s more leading zeros than $f(n)$. By the pumping lemma for finite automata, we can pump the leading zeros and obtain some $(n,m) \neq (n, f(n))$. But $(n, f(n))$ is the only pair with first component n in the graph of f, so we have a contradiction. □

The sequence $(n + \lfloor \log_2 n \rfloor)_{n=0}^{\infty}$ is an example of a 2-regular sequence that is in $O(n)$ but not 2-synchronized, so the converse of Theorem 2.23 does not hold.

Note that a simple way to generalize k-synchronized functions is to allow each component of the domain and codomain to be in a different numeration system. We call such functions *semi-synchronized*, and hope that the numeration systems are understood, since it is usually too cumbersome to list all the different numeration systems. We see some examples of semi-synchronized functions in Chapter 5.

Chapter 3

Decidability in Automatic Sequences

It is possible to answer certain questions about an automatic sequence by mechanically performing transformations on the corresponding automaton. The general process is as follows:

1. Phrase the query as a logical formula in a first-order logical theory.
2. Mechanically translate the query into operations on the automaton for the sequence, as well as automata for the numeration system.
3. Execute the operations to construct an automaton representing the answer to the query.
4. In some cases, it is necessary to interpret the resulting automaton.

We present a formal description of the logical theory, and explain the procedure for converting a logical formula into an automaton.

3.1 Introduction

The logical theory we use is based on the first-order theory of $\mathbb{N}$, which we denote $\mathsf{FO}(\mathbb{N})$. It includes variables over $\mathbb{N}$, logical connectives ($\wedge$, $\vee$, $\Leftrightarrow$, $\neg$, etc.), universal ($\forall$) and existential ($\exists$) quantifiers, and equality comparison ($=$). We represent a word $w \in \Gamma^\omega$ as a finite set of unary predicates, $\{P_a\}_{a\in\Gamma}$, such that $P_a(i)$ is true if $w[i] = a$. These predicates allow us to index specific symbols in w. Hence, our base logical theory is $\mathsf{FO}(\mathbb{N}, \{P_a\}_{a\in\Gamma})$.

If $w \in \Gamma^\omega$ is $\mathcal{N}$-automatic, for some numeration system $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$, we can use automata to decide the theory $\mathsf{FO}(\mathbb{N}, \{P_a\}_{a\in\Gamma})$. That is, there is an algorithm to

decide the truth or falsehood of any sentence in the theory. Depending on the properties of the numeration system, $\mathcal{N}$, we may be able to extend the theory with predicates for comparison, addition, congruence classes modulo any fixed constant and divisibility by powers of k, while maintaining decidability.

Alternatively, we can extend $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ with variables representing sets of integers, predicates for set membership, and quantification over set variables. This gives a *monadic second-order theory* which we denote $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$. The monadic second-order theory is still decidable for many sequences, but it is difficult or impossible to extend with additional predicates. For instance, we can express divisibility in $\mathsf{MSO}(\mathbb{N}, <, +)$, so it is at least as strong as $\mathsf{FO}(\mathbb{N}, <, +, |)$, which is known to be undecidable (see Tarski [50]).

3.2 Deciding First-Order Sentences

Our decision procedures are all based on the idea of representing logical formulas as automata, such that we represent a formula $\phi(a_1, a_2, \ldots, a_n)$ in n free variables with an automaton M on n inputs. Then we can evaluate ϕ at integer values $i_1, \ldots, i_n \in \mathbb{N}$ by feeding corresponding representations $(i_1), \ldots, (i_n)$ into M. As we have seen, there are many ways to represent nonnegative integers as words, but two representations stand out:

- $\mathcal{N}$, the numeration system associated with the automatic sequence, and
- $\mathcal{U} = (\{0, 1\}, 0^*10^*, \langle\cdot\rangle_{\mathcal{U}})$, a unary numeration system where $\langle 0^i 10^j\rangle_{\mathcal{U}} = i$.

This section will focus on the first option. The second option allows us to easily represent subsets of $\mathbb{N}$, so we use it for monadic second-order formulas, discussed in Section 3.3.

Now that we have fixed $\mathcal{N}$ as our numeration system, we need to decide what kind of automaton to use. DFAs are a natural choice, but there are subtle problems that arise with quantifiers. Therefore we will start with ω-automata and then discuss DFAs (and DFAOs) as an optimization.

Theorem 3.1. *Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle)$ be an LSD ideal numeration system with automatic comparison, and let $\mathcal{N}_\omega = (\Sigma, L_\omega, \langle\cdot\rangle_\omega)$ be the corresponding ω-numeration system. Suppose $w \in \Gamma^\omega$ is an $\mathcal{N}$-automatic sequence.*

Given any logical formula $\phi(a_1, \ldots, a_n)$ expressible in $\mathsf{FO}(\mathcal{N}, <, \{P_a\}_{a\in\Gamma})$, define a language of infinite words $\Lambda(\phi) \subseteq L_\omega^n$ where

$$\Lambda(\phi) = \{(x_1, \ldots, x_n) \in L_\omega^n : \phi(\langle x_1\rangle_\omega, \ldots, \langle x_1\rangle_\omega)\}.$$

Then $\Lambda(\phi)$ is ω-regular, and we can effectively construct the corresponding ω-automaton.

Proof. Recall that a formula is defined recursively as

1. one of the atomic predicates ($<$ or P_a),
2. a negated formula ($\neg\phi$),
3. a pair of formulas joined by a binary connective ($\phi \vee \psi$, $\phi \wedge \psi$), or
4. a quantified formula ($\forall x\phi(x)$, $\exists x\phi(x)$).

We induct on the size of the formula, so we may assume that $\Lambda(\psi)$ is ω-regular for any subformula ψ.

1. Comparison is automatic (by assumption) so $\Lambda(x < y)$ is ω-regular (see Theorem 2.3). Similarly, since w is $\mathcal{N}$-automatic, we can construct a DFA for the language

$$L_a := \{x \in L : w[\langle x \rangle] = a\}$$

   for all $a \in \Gamma$, and then extend these languages to ω-regular languages for $\Lambda(P_a) \subseteq \Sigma^\omega$.

2. Let ϕ be a logical formula with n free variables. Then $(x_1, \ldots, x_n) \in L_\omega^n$ is a member of $\overline{\Lambda(\phi)}$ if and only if it belongs to $\Lambda(\neg\phi)$. Therefore

$$\Lambda(\neg\phi) = \overline{\Lambda(\phi)} \cap L_\omega^n,$$

   so $\Lambda(\neg\phi)$ is ω-regular.

3. Let ϕ_1 and ϕ_2 be logical formulas in the theory having m_1 and m_2 free variables respectively. If ϕ_1 and ϕ_2 have the same set of $m_1 = m_2$ free variables, then

$$\Lambda(\phi_1 \vee \phi_2) = \Lambda(\phi_1) \cup \Lambda(\phi_2)$$
$$\Lambda(\phi_1 \wedge \phi_2) = \Lambda(\phi_1) \cap \Lambda(\phi_2)$$

   and we are done. If they have different sets of free variables, then we need to think of f_1 and f_2 as formulas over a common set of n variables, and promote $\Lambda(f_1)$ and $\Lambda(f_2)$ to languages on n inputs. To this end, we introduce codings

$$h_1 \colon \Sigma^{n\times\omega} \to \Sigma^{m_1\times\omega}$$
$$h_2 \colon \Sigma^{n\times\omega} \to \Sigma^{m_2\times\omega}$$

   where h_i drops all inputs that do not correspond to free variables in ϕ_i, and reorders the remaining variables appropriately. For example, given $\phi_1(x, y)$ and $\phi_2(z, x)$, we would define $h_1(x, y, z) = (x, y)$ and $h_2(x, y, z) = (z, x)$.

Then it is not hard to see that

$$\Lambda(\phi_1 \vee \phi_2) = L_\omega^n \cap \left(h_1^{-1}(\Lambda(\phi_1)) \cup h_2^{-1}(\Lambda(\phi_2))\right)$$
$$\Lambda(\phi_1 \wedge \phi_2) = L_\omega^n \cap h_1^{-1}(\Lambda(\phi_1)) \cap h_2^{-1}(\Lambda(\phi_2)).$$

Other binary connectives can be constructed from $\vee$, $\wedge$ and $\neg$.

4. We consider only $\exists x \phi(x)$, since $\forall x \phi(x)$ is equivalent to $\neg \forall x \neg \phi(x)$. Suppose ϕ is a formula over n variables, including x, the quantified variable. Let $h_x \colon \Sigma^{n \times \omega} \to \Sigma^{(n-1) \times \omega}$ be a coding that drops the input corresponding to x, the quantified variable. Then

$$\begin{aligned} h_x(\Lambda(\phi)) &= \{(y_1, \ldots, y_{n-1}) \in L_\omega^{n-1} : \exists x \in L_\omega \text{ such that } (x, y_1, \ldots, y_{n-1}) \in \Lambda(\phi)\} \\ &= \{(y_1, \ldots, y_{n-1}) \in L_\omega^{n-1} : \exists x \in L_\omega \text{ such that } \phi(\langle x \rangle_\omega, \langle y_1 \rangle_\omega, \ldots, \langle y_{n-1} \rangle_\omega)\} \\ &= \{(y_1, \ldots, y_{n-1}) \in L_\omega^{n-1} : \exists x \, \phi(x, \langle y_1 \rangle_\omega, \ldots, \langle y_{n-1} \rangle_\omega)\} \\ &= \Lambda(\exists x \, \phi(x)). \end{aligned}$$

It follows that $\Lambda(\exists x \, \phi(x))$ is ω-regular.

□

Let us see examples of what is possible in the theory $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a \in \Gamma})$.

- We can compare two positions of w. The predicate is

$$\bigvee_{a \in \Gamma} (P_a(i) \wedge P_a(j))$$

which we will abbreviate to $w[i] = w[j]$.

- The successor predicate, $y = x + 1$, can be expressed as

$$(x < y) \wedge (\forall z \, (z \leq x) \vee (y \leq z)).$$

That is, $x < y$ and there is no z strictly between them. Also note that zero is the unique $z \in \mathbb{N}$ satisfying $(\forall x \, (x \geq z))$. With successor and zero, we can express any constant $c \in \mathbb{N}$, and we can perform addition by any fixed constant. For convenience, we will assume these operations are primitive operations in the theory.

- We can take advantage of addition by constants to construct any tail $w[c..\infty]$ of w. Apply Theorem 3.1 to the formulas $P_a(i+c)$ for each $a \in \Gamma$. This collection of predicates corresponds to the word $w[c..\infty]$; recall that Theorem 1.13 allows us to assemble regular language into a DFAO. Moreover, it proves that if w is $\mathcal{N}$-automatic then so is $w[c..\infty]$. It is possible to prove this more directly and constructively for morphic sequences; see [6, Theorem 7.6.1, p. 228].

- Single symbol insertion and deletion are also expressible with formulas. Given predicates $\{P_a\}_{a\in\Gamma}$, we create a new set of predicates $\{P'_a\}_{a\in\Gamma}$ where

$$P'_a(n) := (n < i_{\text{delete}} \wedge P_a(n)) \vee (n \geq i_{\text{delete}} \wedge P_a(n+1))$$

  to delete $w[i_{\text{delete}}]$. To insert a at position i_{insert}, we let

$$P'_a(n) := (n < i_{\text{insert}} \wedge P_a(n)) \vee (n = i_{\text{insert}}) \vee (n > i_{\text{insert}} \wedge P_a(n-1))$$

  and

$$P'_b(n) := (n < i_{\text{insert}} \wedge P_b(n)) \vee (n > i_{\text{insert}} \wedge P_b(n-1))$$

  for all $b \neq a$. This easily generalizes to any finite sequence of edits.

- Given a word $x \in \Gamma^*$, define the predicate $Q_x(i)$, which is true if there is an occurrence of x in w, starting at position i. Formally, the predicate is

$$Q_x(i) = P_{x[0]}(i) \wedge P_{x[1]}(i+1) \wedge \cdots \wedge P_{x[n-1]}(i+n-1)$$

  where n is the length of x. Therefore we can detect occurrences of a given finite word.

- For any predicate Q in $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$, we can decide whether the set $\{i \in \mathbb{N} : Q(i)\}$ is empty or infinite, using the predicates $\exists i\, Q(i)$ and $\forall i(\exists j(j \geq i \wedge Q(j))$ respectively. We can also compute $\min\{i : Q(i)\}$ and $\max\{i : Q(i)\}$, if they exist.

- We can express *counting quantifiers* in $\mathsf{FO}(\mathbb{N}, <)$. A counting quantifier is an existential quantifier with an additional restriction on the number of solutions. One special case is the *uniqueness quantifier*, written $\exists!$, which is satisfied if there is a unique solution. We also have "there exist exactly n solutions" ($\exists^n$), "there exist at least n solutions" ($\exists^{\geq n}$) and "there exist infinitely many" ($\exists^\infty$). We express these as follows:

$$\begin{aligned}
\exists^{\geq n} x\, \phi(x) &:= \exists x_1, \ldots, x_n\, (x_1 < x_2 < \cdots < x_n) \wedge \phi(x_1) \wedge \phi(x_2) \wedge \cdots \wedge \phi(x_n) \\
\exists^{n} x\, \phi(x) &:= (\exists^{\geq n} x\, \phi(x)) \wedge (\neg \exists^{\geq n+1} x\, \phi(x)) \\
\exists^{\infty} x\, \phi(x) &:= \forall x_0 \exists x\, (x \geq x_0) \wedge \phi(x)
\end{aligned}$$

  where $n \geq 1$ is a constant.

3.2.1 Additional Operations

It is very easy to add new operations to the theory. Recall that the proof of Theorem 3.1 is a structural induction on formulas. That is, for each atomic formula or primitive operation in $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$, there is a corresponding case in the proof. If we extend the theory with a new atomic formula, ϕ, it suffices to give an ω-automaton for $\Lambda(\phi)$. To add a new operator (for instance, something that binds a variable), we must show that we can effectively construct the necessary automaton, given automata for each subformula.

Addition

We saw earlier that it is possible to use comparison and existential quantifiers to add a constant. That is, for any constant $c \in \mathbb{N}$, the binary predicate $x + c = y$ is in $\mathsf{FO}(\mathbb{N}, <)$. Here we consider full addition, embodied by the ternary predicate $x+y = z$. By definition, the language $\Lambda(x + y = z)$ is regular if and only if addition is automatic in the ambient numeration system.

The first-order theory of the natural numbers with addition, $\mathsf{FO}(\mathbb{N}, <, +)$, is known as *Presburger arithmetic*. Presburger showed that this theory is decidable (see [41] for a translation). Later, Büchi [12] gave an automaton-based proof that Presburger arithmetic is decidable. The same result follows from our Theorem 3.1 with a numeration system $\mathcal{N}$ where addition is automatic.

Addition is a powerful operation that is essential for many interesting queries about automatic sequences. Note that when we write $\phi(i + j)$, we actually mean $\exists k\,(i + j = k) \wedge \phi(k)$, but we will write the former for readability. [1] Let w be an $\mathcal{N}$-automatic sequence.

- We can use repeated addition to express multiplication by a constant,

$$dn = \overbrace{n + \cdots + n}^{d \text{ times}},$$

  and we can divide by a constant by finding $q, r \in \mathbb{N}$ such that $0 \leq r < d$ and $n = qd + r$.

[1] In some numeration systems (e.g., base k), we can use a transducer to compute $i+j$ on the fly, without a new variable k and the subsequent existential quantification.

- Given a starting position i and length n of a subword in our sequence, we can compute the endpoint, $j := i + n - 1$. Similarly, given a startpoint i and endpoint j of a subword, we can compute the length, $n := j - i + 1$.

- We can test for equality of two subwords, $w[i..i + n - 1]$ and $w[j..j + n - 1]$, using the following predicate.

$$(\forall k < n\,(w[i + k] = w[j + k]))$$

  This is an extremely common test in more complicated predicates, so we abbreviate it as $w[i..i + n - 1] = w[j..j + n - 1]$.

- We can test if $w[i..i + n - 1]$ is lexicographically less than $w[j..j + n - 1]$ as follows.

$$(\exists k < n\,(w[i + k] < w[j + k]) \wedge (w[i..i + k - 1] = w[j..j + k - 1]))$$

- A subword $w[i..i+n-1]$ is a palindrome if and only if $w[i..i+n-1] = w[i..i+n-1]^R$. We can reverse subwords with arithmetic, so

$$(\forall k < n\,(w[i] = w[i + n - k - 1]))$$

  accepts is true iff $w[i..i + n - 1]$ is a palindrome.

- We say an occurrence of a subword is *novel* if there is no earlier occurrence of the same subword. We can express this as a predicate:

$$(\forall j < i\,(w[i..i + n - 1] \neq w[j..j + n - 1])).$$

  The number of novel factors of length n is the subword complexity. For a k-automatic sequence w, it follows that the subword complexity is k-regular. In [30], we show that the subword complexity is actually k-synchronized.

- Recall the *appearance function* $A_w \colon \mathbb{N} \to \mathbb{N}$ where $A_w(n)$ is the length of the shortest prefix of w that contains all subwords of length n. The following formula in terms of m and n is true if $w[0..m - 1]$ contains all subwords of length n.

$$R(m, n) := (\forall i(\exists j \leq (m - n)\,(w[i..i + n - 1] = w[j..j + n - 1])))$$

  Then the predicate

$$R(m, n) \wedge (\forall m_0\,(R(m_0, n) \Longrightarrow m_0 > m))$$

  is true if $m = A_w(n)$. The recurrence function and condensation function are similarly expressible as predicates.

- Recall that a word x has period p if $x[j] = x[j+p]$ for all possible j. In other words, $w[i..i+n-1]$ has period p if $w[i..i+n-p-1] = w[i+p..i+n-1]$, which is a predicate. We leave it as an exercise to express "p is the maximum period of $w[i..i+n-1]$" and "p is a dth power" (for fixed d) as predicates.

- Suppose $x \in \Gamma^*$ is a word. Then $y \in \Gamma^*$ is a *conjugate* of x if there exists $u, v \in \Gamma^*$ such that $x = uv$ and $y = vu$. We say x is a *Lyndon word* if it is not of the form y^d, and it is lexicographically less than all its conjugates. We can test if subwords of w are Lyndon words using predicates. With some additional work, Goč and Shallit [31] show how to compute the *Lyndon factorization* of w. That is, they divide w into nonempty Lyndon words $w = w_0 w_1 w_2 \cdots$ such that $w_0 < w_1 < w_2 < \cdots$.

- We can test whether w is periodic ($\exists p(w[0..\infty] = w[p..\infty])$) or ultimately periodic ($\exists i, p(w[i..\infty] = w[i+p..\infty])$) using predicates.

- The *shift orbit closure* of an infinite word $w \in \Gamma^\omega$ is the set $S \subseteq \Gamma^\omega$ where

$$S := \{x \in \Gamma^\omega : \text{every subword of } x \text{ is a subword of } w\}.$$

  We can compute the lexicographically least element $z \in S$ of the orbit closure, using the following observation. The prefix $z[0..n]$ is a subword of w of length $n+1$, since $z \in S$. If there is some factor $w[i..i+n]$ that is lexicographically less than $z[0..n]$, then $w[i..\infty]$ is lexicographically less than z. Therefore, $z[n]$ is the last character of the lexicographically least subword of length $n+1$ in w. We can express a family of predicate $\{Q_a\}_{a\in\Gamma}$ such that $Q_a(n)$ is true if and only if $z[n] = a$.

$$Q_a(n) := (\exists i\,(w[i+n] = a) \wedge (\forall j(w[i..i+n] \leq w[j..j+n])))$$

  If w is $\mathcal{N}$-automatic, it follows that z is $\mathcal{N}$-automatic. See Problem 6.7 for an open problem about the shift orbit closure.

Arithmetic Sequence Predicates

A natural predicate, which unfortunately appears to be inexpressible in $\mathsf{FO}(\mathbb{N}, <)$, is the statement $x \equiv y \pmod d$, for d a constant. We can express it in $\mathsf{FO}(\mathbb{N}, <, +)$ as

$$\exists s, t \text{ such that } x + ds = y + dt$$

but we have seen that addition places non-trivial conditions on $\mathcal{N}$. It turns out that we can extend the first-order theory (and Theorem 3.1) with binary predicates $\{M_d\}_{d=2}^{\infty}$ where $M_d(x, y) := (x \equiv y \pmod d)$, without any additional assumptions.

Theorem 3.2. *Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ be an ideal numeration system such that comparison is automatic. Then any ultimately periodic sequence $w \in \Gamma^\omega$ is $\mathcal{N}$-automatic.*

Proof. We present a proof below. See Lecomte and Rigo [35] for an alternate proof.

Let $w \in \Gamma^\omega$ be an ultimately periodic sequence. We will assume without loss of generality that w is periodic, since we can make a finite set of changes to an $\mathcal{N}$-automatic sequence (using predicates, or other constructions). Suppose that w has period d. Then each of the predicates $\{P_a\}_{a\in\Gamma}$ is a finite union of congruence classes modulo d. It suffices to show that we can construct an automaton for the predicate $i \equiv 0 \pmod{d}$, since the other congruence classes follow using shifts (addition by constants).

Note that for a representation $x \in L$,

$$\begin{aligned}\langle x\rangle_{\mathcal{N}} &= \#\{i \in \mathbb{N} : i < \langle x\rangle_{\mathcal{N}}\}\\ &= \#\{y \in L : (y, x) \in L_<\}\\ &= \#\text{ of accepting paths in } L_< \text{ with second coordinate } x.\end{aligned}$$

We can count accepting paths in $L_<$ with matrices. Specifically, there exist vectors $u, v \in \mathbb{N}^m$ and a homomorphism $h\colon \Sigma^* \to \mathbb{N}^{m\times m}$ (to the monoid of matrices under multiplication) such that the number of accepting paths is $u^T h(x)v$.

We want to compute $u^T h(x)v \bmod d$. Since the map $n \mapsto n \bmod d$ is a ring homomorphism, and matrix multiplication uses only ring operations, we have

$$u^T h(x)v \bmod d = \overline{u}^T \overline{h}(x)\overline{v}$$

where $\overline{u}, \overline{v} \in (\mathbb{Z}/d\mathbb{Z})^m$ and $\overline{h}\colon \Sigma^* \to (\mathbb{Z}/d\mathbb{Z})^{m\times m}$ are the images of u, v and h respectively, modulo d. Now we observe that $(\mathbb{Z}/d\mathbb{Z})^{m\times m}$ is a finite monoid, having only d^{m^2} elements. It follows that the function

$$f(i) = \begin{cases} i \bmod d, & \text{if } x \in L,\\ \varnothing, & \text{if } x \notin L,\end{cases}$$

is $\mathcal{N}$-automatic, which concludes the proof. □

k-adic Valuation

With base-k numbers, certain properties of an integer i are apparent directly from its string representation. For instance, divisibility of i by powers of k, or the largest power of k less than i.

Definition 3.3. Let $k \geq 2$ be an integer. The *k-adic power* of $n \in \mathbb{N}$, denoted $V_k(n) \in \mathbb{N}$, is the largest power of k that divides n.

In other words, if $(n)_k$ has t trailing zeros then $V_k(n) = k^t$. This makes it easy to compute $V_k(n)$ using an automaton.

Proposition 3.4. *Fix an integer $k \geq 2$. The language*

$$\{(n, V_k(n))_k : n \geq 0\} \subseteq \Sigma_k^*$$

is regular.

It follows that the first-order theory $\mathsf{FO}(\mathbb{N}, <, +, V_k)$ is decidable.

Using Theorem 3.1, with the appropriate extensions for $\mathsf{FO}(\mathbb{N}, <, +, V_k)$, any formula ϕ in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$ is true on a k-automatic set. Remarkably, the converse is true by a theorem of Bruyère [10].

Theorem 3.5. *Let $T \subseteq \mathbb{N}^m$ be a k-automatic set. Then there exists a formula ϕ in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$ such that $\phi(a_1, \ldots, a_m)$ is true if and only if $(a_1, \ldots, a_m) \in T$.*

Proof. See [10, 11] □

We give an example taken from [11]; we express the Thue-Morse sequence, $\mathbf{t}$, as a predicate in $\mathsf{FO}(\mathbb{N}, <, +, V_2)$. Recall that we can compute $\mathbf{t}[i]$ with the automaton in Figure 2.2, which computes the sum of the bits in the input, modulo 2. We introduce an operation $D_2 \colon \mathbb{N} \to \mathbb{N}$ where $D_2(n)$ is the natural number obtained by replacing every other 1 in $(n)_2$ with a 0, starting with the leading one. For instance, $D_2(13865) = 4616)$ because

$$(4616)_2 = 01001000001000$$
$$(13865)_2 = 11011000101001.$$

We claim that we can express the predicate $D_2(n) = m$ in $\mathsf{FO}(\mathbb{N}, <, +, V_2)$, and may therefore use D_2 to construct a predicate for Thue-Morse. In particular, we observe that $V_2(n)$ gives the position of the least significant 1 in n, and the least significant 1 of $D_2(n)$ is the same if and only if n has an even number of zeros. Hence, $\mathbf{t}[n] = 0$ if and only if $V_2(D_2(n)) = V_2(n)$, and therefore it suffices to show that we can express $D_2(n) = m$ in $\mathsf{FO}(\mathbb{N}, <, +, V_2)$.

To express $D_2(n) = m$, we introduce yet another predicate, $\in_2 (m, n)$. Define $\in_2 (m, n)$ to be true if m is a power of 2, and m is a term in the binary expansion of n. For instance, $\in_2 (32, 13865)$ because

$$(32)_2 = 00000000100000$$
$$(13865)_2 = 11011000101001.$$

Assuming m is a power of 2 (which we can express), observe that m is a term in the binary expansion of n if and only if

$$(\exists i, j((V_2(i) > m) \wedge (j < m) \wedge (i + m + j = n))).$$

This is based on decomposing n into m, more significant bits (i), and less significant bits (j). Given $\in_2 (m, n)$, the predicate $D_2(n) = m$ is expressible because it holds if and only if the following three conditions hold.

- Every power of 2 that appears in m also occurs in n. That is,

$$(i = V_2(i)) \wedge \in_2 (i, m) \Longrightarrow \in_2 (i, n).$$

- For any two consecutive powers of 2 in n, exactly one of the powers appears in m. This is expressible with $\in_2$ and V_2.

- The leading power of 2 in n does not appear in m. We express this as

$$\forall i \Big(\big((i = V_2(i)) \wedge \in_2 (i, n) \wedge (\forall j > i \, (j = V_2(j) \Longrightarrow \neg \in_2 (j, n))) \big) \Longrightarrow \neg \in_2 (i, m) \Big).$$

We conclude that $\mathbf{t}[n] = 1$ can be expressed in $\mathsf{FO}(\mathbb{N}, <, +, V_2)$, although the conversion is by no means trivial.

For our purposes, the main consequence of Theorem 3.5 is that we cannot meaningfully extend Theorem 3.1 with predicates beyond $\mathsf{FO}(\mathbb{N}, <, +, V_k)$, since any predicate corresponds to a k-automatic set, and therefore can already be expressed in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$. Even the predicates $\{P_a\}_{a \in \Gamma}$ associated with a k-automatic sequence can be expressed in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$.

If $\mathcal{N}$ is some numeration system other than base-k representation, then we can still sometimes define a function $V_{\mathcal{N}}$ analogous to V_k. For instance, if $\mathcal{N}$ is the Fibonacci numeration system, then $V_{\mathcal{N}}(n)$ is awkwardly defined as the highest Fibonacci number F_k such that n can be written as a sum of $\{F_j\}_{j \geq k}$. Surprisingly, this definition is useful.

- We will not formally define *local period*, but it is a measure of the periodicity of a word at a given position. Shallit proved that the local period at position n in the Fibonacci word is closely related to $V_{\mathcal{N}}(n)$, where $\mathcal{N}$ is the Fibonacci numeration system. This relationship was subsequently generalized to characteristic Sturmian words and Ostrowski representation by Schaeffer [48].

- Kalle Saari showed in his Ph.D. thesis [47] that all length-n factors of the Fibonacci word are bordered unless n is a Fibonacci number. We can express "n is a Fibonacci number" as the formula $V_{\mathcal{N}}(n) = n$. Since addition is possible in Fibonacci representation, we can express the entire theorem as a predicate,

  $$(\forall n\, (\forall i\, B(i,n)) \vee (V_{\mathcal{N}}(n) = n)$$

  where $B(i,n)$ represents the predicate "$w[i..i+n-1]$ is bordered", and then mechanically prove the theorem.

- The Zeckendorf array $Z\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (see [34]) is defined such that the jth column is the increasing sequence of all natural numbers n satisfying $V_{\mathcal{N}}(n) = F_{j+1}$. This has a number of interesting properties.
  - Every natural number occurs exactly once in the array.
  - Every row of the Zeckendorf array is a generalized Fibonacci sequence. That is, it a sequence $(x_n)_{n=0}^{\infty}$ satisfying $x_n = x_{n-1} + x_{n-2}$ for all $n \geq 2$.
  - For any positive generalized Fibonacci sequence $(x_n)_{n=0}^{\infty}$, some suffix $(x_n)_{n=n_0}^{\infty}$ is a row in Z.

3.2.2 Decidability using DFAs and DFAOs

Observe that in Theorem 3.1, we use a handful of closure properties for ω-regular languages. Specifically,

- closure under intersection, union and complement,
- closure under morphic image, and
- closure under inverse morphism.

Regular languages easily satisfy all these closure properties, and the languages corresponding to the atomic formulas $x < y$ and $P_a(x)$ are regular. Indeed, most of the constructions in the proof of Theorem 3.1 carry over directly to regular languages without modification, but existential quantification is not so easy.

Recall that in the proof of Theorem 3.1, we have $\Lambda(\exists x \phi(x)) = h(\Lambda(\phi))$, where h is a morphism that drops the input corresponding to the variable x. Suppose for illustration that ϕ has two free variables, x and y. By definition,

$$h(\Lambda(\phi)) = \{y \in \Sigma_k^\omega : \exists x \in \Sigma_k^\omega \text{ such that } (x, y) \in \Lambda(\phi)\}.$$

Note that the quantification $\exists x$ is over infinite words, not integers. But every natural number is represented by some infinite word, so we are effectively quantifying over natural numbers.

Suppose that instead we use DFAs to accept a regular language $\Lambda(\phi) \subseteq (\Sigma_k \times \Sigma_k)^*$. As in the infinite case, $\Lambda(\phi)$ is a set of pairs of representations (x, y) such that ϕ is true on the corresponding pair of integers. Recall that we accept multiple inputs with finite automata by merging the inputs into a single word over a larger alphabet, padding the words to the same length as necessary. As a result, if we use a morphism h to drop the x variable, we obtain

$$\{y \in \Sigma_k^* : \exists x \in \Sigma_k^{|y|} \text{ such that } (x, y) \in \Lambda(\phi)\}.$$

That is, we only quantify over representations with the same number of digits as y.

For example, consider the predicate $\exists x\,(x = ky + 1)$ with base-k representations. The predicate is clearly always true, but the representation for the witness x is always one digit longer than the representation of y, so the naïve delete-and-determinize procedure produces an automaton which *rejects* all representations without leading zeros. On the other hand, it accepts any representation with at least one leading zero, so the automaton is not consistent over all representations of a given number. Even worse, the automaton is incorrect on all canonical representations (i.e., the representation without leading zeros).

We will show how to solve the leading zero problem, allowing us to resolve existential quantifiers, and hence decide first-order logical predicates using DFAs. Observe that the automaton produced by the naïve delete-and-determinize procedure is *eventually correct* in the sense that, if there does exist x such that $\phi(x, y)$ then the automaton accepts all sufficiently long representations for y, and if there does not exists x such that $\phi(x, y)$ then the automaton accepts no representation of y. In other words, if the automaton accepts some representation $y0^*$ (assuming LSD first), then there exists x such that $\phi(x, y)$ and it should accept y. If L is the language accepted by the automaton, then we want to compute an automaton for

$$\begin{aligned} L' &= \{w \in \Sigma^* : w0^* \cap L \text{ nonempty}\} \\ &= \{w \in \Sigma^* : \exists y \in 0^* \text{ such that } wy \in L\} \\ &= L/0^*, \end{aligned}$$

a right quotient of L. Therefore L' is regular and computable, although in this case it is especially efficient to compute.

Theorem 3.6. *Let $T = (\Sigma, Q, \delta, q_0, F)$ be a DFA for a language $L = L(T)$, and suppose $0 \in \Sigma$ is a symbol. There is an algorithm to compute the DFA for $L/0^*$ in linear time (in the size of T).*

Proof. Consider a directed graph on Q where the edges are the reversed 0 transitions. We mark all nodes that are reachable from the set F. The set of marked nodes, $F' \subseteq Q$, is the set of accepting nodes for the new automaton $T' = (\Sigma, Q, \delta, q_0, F')$.

It is clear that $L(T') = L(T)/0^*$, because the accepting set F' is precisely the set of nodes that have a path of 0s to some state in F. It is equally clear that the algorithm is linear time. Either we use a simultaneous breadth-first search (BFS) from all nodes in F, or perform a series of breadth-first or depth-first searches (at each node in F) which terminate at nodes that we have already marked as reachable. □

Therefore, we compute $\Lambda(\exists x \phi(x))$ by determinizing $h_x(\Lambda(\phi))$ and applying the quotient algorithm described above. In full generality, $\Lambda(\exists x \phi(x))$ may be over $(\Sigma \times \cdots \times \Sigma)^*$, so we may take a quotient by $(0, 0, \ldots, 0)^*$ instead of 0^*, but the idea is the same.

We can use DFAs to decide first-order sentences and to show that the subset of $\mathbb{N}^d$ satisfying a predicate P is $\mathcal{N}$-automatic. Goč's implementation [29] is based on DFAs instead of ω-automata because the operations are simpler to implement, and the resulting automata are often easier to understand.

3.2.3 Complexity

Suppose we are given a formula ϕ in the first order theory. Assuming we use DFAs for our implementation, we can compute an upper bound for each subformula of ϕ, starting with the state complexities of the automata for $\{P_a\}_{a \in \Gamma}$ and the other automatic predicates, and working our way up using known state complexity bounds for DFA operations. The most expensive operations are existential and universal quantification, requiring a determinization and a potentially exponential blowup in state complexity. We can perform multiple quantifiers *of the same type* at once, so our upper bound is (roughly) a tower of exponentials where the height of the tower is related to the number of quantifier alternations.

However, in practice we have never encountered performance close to the predicted $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$ upper bound for complicated predicates. In fact, the limiting factor in our experiments is often k, the size of the alphabet, and the number of free variables m, because

a predicate with m free variables corresponds to a DFA on Σ_k^m, with k^m transitions out of every state. It is not unusual for a predicate to require 5 or even 6 simultaneous free variables. In the case of the (13-automatic) Leech word [36], defined as the fixed point of

$$\begin{aligned} 0 &\mapsto 0121021201210 \\ 1 &\mapsto 1202102012021 \\ 2 &\mapsto 2010210120102, \end{aligned}$$

this leads to $13^5 = 371293$ or $13^6 = 4826809$ transitions out of every state. With a bit of patience, we can resolve all but the most complicated predicates on the Leech word. On the other hand, the implementation quickly exhausts the available memory on Keränen's 85-automatic sequence [33], even for predicates of moderate complexity.

3.3 Deciding Monadic Second-Order Sentences

Second-order logic extends first-order logic with k-ary relations on the domain. In our case the domain is $\mathbb{N}$, and a k-ary relation is simply a subset of $\mathbb{N}^k$. Monadic second-order logic is a restriction of second-order logic that allows quantification only over 1-ary relations (i.e., subsets of $\mathbb{N}$). Hence, $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ extends $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ with set variables, quantification over set variables, and set membership testing. We also include set equality comparison, subset comparison, set intersection, union and set difference, since they can be expressed in $\mathsf{MSO}(\mathbb{N}, <)$ and it is convenient to have notation for them.

We can decide sentences in $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ using automata, but not with the same variable representation as first-order sentences. First, let us see a decision procedure for $\mathsf{MSO}(\mathbb{N}, <)$. This is an old result, with an automata-theoretic proof due to Büchi [12] in 1962.

Theorem 3.7. *Define a function $\chi\colon 2^{\mathbb{N}} \to \{0,1\}^\omega$ such that $\chi(X)$ is an infinite binary word where*

$$\chi(X)[i] = \begin{cases} 0, & \text{if } i \notin X; \\ 1, & \text{if } i \in X. \end{cases}$$

We abuse notation and overload χ so that $\chi(i) = \chi(\{i\})$ for $i \in \mathbb{N}$.

Suppose $\phi(a_1, \ldots, a_n, A_1, \ldots, A_m)$ is a formula in $\mathsf{MSO}(\mathbb{N}, <)$ with free integer variables

$a_1, \ldots, a_n$ and free set variables $A_1, \ldots, A_m$. Define $\Lambda_\infty(\phi) \subseteq (\{0,1\}^{m+n})^\omega$ where

$$\Lambda_\infty(\phi) = \{(\chi(x_1), \ldots, \chi(x_n), \chi(X_1), \ldots, \chi(X_m)) : x_1, \ldots, x_n \in \mathbb{N} \\ X_1, \ldots, X_m \subseteq \mathbb{N} \\ \phi(x_1, \ldots, x_n, X_1, \ldots, X_m)\}.$$

Then $\Lambda_\infty(\phi)$ is ω-regular and we can explicitly construct the corresponding ω-automaton.

Proof. Recall that a formula is one of the following:

- an atomic predicate ($x = y$, $x < y$ or $x \in X$),
- a negated formula ($\neg\phi$),
- two formulas joined by a binary connective ($\phi \vee \psi$ or $\phi \wedge \psi$), or
- a quantified formula ($\exists x \phi(x)$ or $\forall x \phi(x)$).

Let us consider the automatic formulas first. Suppose X, Y are subsets of $\mathbb{N}$. Clearly $X \subseteq Y$ if and only if $(\chi(X), \chi(Y))$ is in $\{(0,0),(0,1),(1,1)\}^\omega$, which is an ω-regular language. Similarly, $X = Y$ if and only if $(\chi(X), \chi(Y))$ is in the ω-regular language $\{(0,0),(1,1)\}^\omega$. Since we represent integer variables as singleton sets, this shows that $\Lambda_\infty(x = y)$ and $\Lambda_\infty(x \in X)$ are ω-regular. It is also clear that $i < j$ if and only if $(\chi(i), \chi(j))$ is in $(0,0)^*(1,0)(0,0)^*(0,1)(0,0)^\omega$, an ω-regular language. Therefore, for any atomic formula ϕ, the ω-language $\Lambda_\infty(\phi)$ is ω-regular.

The proofs for negation, binary connectives and quantification are very similar to the analogous operations in first-order logic, as proved in Theorem 3.1.

- We negate a formula ϕ by taking the complement of $\Lambda_\infty(\phi)$ and intersecting with the universe of all inputs, $\mathcal{U}$, which restricts integer inputs to be of the form $\chi(i)$ for $i \in \mathbb{N}$. That is,
$$\Lambda_\infty(\neg\phi) = \overline{\Lambda_\infty(\phi)} \cap \mathcal{U}.$$

- Conjunction and disjunction correspond to intersection and union (in each case, followed by intersection $\mathcal{U}$), but the difficulty is that the set of free variables in the two formulas, ϕ, ψ, may differ. With appropriate morphisms, h_1 and h_2, and a universe $\mathcal{U}$ for the combined set of free variables, we have
$$\Lambda_\infty(\phi \wedge \psi) = \Lambda_\infty(h_1^{-1}(\phi)) \cap \Lambda_\infty(h_2^{-1}(\psi)) \cap \mathcal{U}$$

- We observe that $\forall x \phi(x)$ is equivalent to $\neg \exists x \neg \phi(x)$, so we can avoid universal quantification. For existential quantification, define a morphism h_X that drops the input corresponding to variable X. Then

$$\Lambda_\infty(\exists X \phi(X)) = h_X(\Lambda_\infty(\phi))$$

  is an ω-regular language. Here, X suggests a set variable, but it is true if X is an integer variable.

□

Elgot and Rabin extended Büchi's automata-theoretic proof to $\mathsf{MSO}(\mathbb{N}, <, P)$, for certain predicates P, including morphic words. It turns out that for $X \subseteq \mathbb{N}$ a set of integers, the ω-language $\chi(X)$ is ω-regular if and only if X is ultimately periodic. Hence, for an aperiodic sequence corresponding to predicates $\{P_a\}_{a \in \Gamma}$, the set $\Lambda_\infty(P_a)$ is not ω-regular. Since we cannot express the predicates $\{P_a\}_{a \in \Gamma}$ directly as automata (as we did in the first-order theory), Elgot and Rabin take an indirect approach.

Given a sentence ϕ in $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a \in \Gamma})$, we replace each occurrence of P_a with a variable X_a, creating a logical formula ϕ' with $|\Gamma|$ free variables. By Theorem 3.7, we can construct an ω-automaton M for $\Lambda_\infty(\phi')$. Elgot and Rabin show how to compute $M(w)$, for certain infinite words $w \in \Gamma^\omega$. We assume a similar result for $\mathcal{N}$-automatic words, Theorem 3.8, without proof, because it follows from Corollary 3.15. See Section 3.4.2 for further details.

Theorem 3.8. *Let $w \in \Gamma^\omega$ be an $\mathcal{N}$-automatic sequence, where $\mathcal{N}$ is an ideal numeration system with lexicographic comparison. Let M be an ω-automaton over the input alphabet Γ. Then there is an algorithm that decides whether M accepts w.*

Corollary 3.9. *Let $w \in \Gamma^\omega$ be an $\mathcal{N}$-automatic sequence, where $\mathcal{N}$ is an ideal numeration system with lexicographic comparison. Then the theory $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a \in \Gamma})$ is decidable.*

Note that the decision procedure is for sentences only, so we can only answer yes/no questions. For example, we can decide if a fixed symbol $b \in \Gamma$ occurs at an odd position (any odd position) in the word. The following sentence checks this condition by constructing sets X_{odd} and X_{even} that contain the odd and even numbers respectively, and then looking for a position $i \in X_{odd}$ such that $w[i] = b$.

$(\exists X_{odd}, X_{even} \subseteq \mathbb{N}$
$\quad (0 \in X_{even}) \wedge$

$$
\begin{aligned}
&(\forall a \in \mathbb{N}\,(a \in X_{odd} \Leftrightarrow a \notin X_{even}))\wedge\\
&(\forall b \in \mathbb{N}\,(b \in X_{odd} \Leftrightarrow b+1 \in X_{even}))\wedge\\
&(\exists i \in \mathbb{N}\\
&\qquad (i \in X_{odd}) \wedge P_b(i)))
\end{aligned}
$$

Similarly, we can write a formula ϕ in $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ with a free variable i such that $\phi(i)$ is true if i is odd and $w[i] = b$.

Unfortunately, unlike the decision procedure for first-order logic, this algorithm does not construct automata for arbitrary formulas. In first-order logic, we can apply the procedure to a formula ϕ, and then use the resulting automaton to evaluate $\phi(x_1, \ldots, x_n)$ for any $x_1, \ldots, x_n \in \mathbb{N}$ we choose. For a monadic second-order formula ϕ, we can only apply Corollary 3.9 if we first evaluate $\phi(x_1, \ldots, x_n)$ by substituting the x_i's into ϕ to obtain a sentence. Also note that we cannot always substitute a set variable into an MSO formula, since there are only countably many logical formulas and uncountably many subsets of $\mathbb{N}$.

Assuming some conditions on the numeration system of our automatic sequence w, there is a procedure that takes a formula ϕ in $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ and generates an ω-automaton for $\Lambda(\phi)$, as long as ϕ has no free set variables. We state and prove this result in Theorem 3.17, but first we need to define the operation that makes it possible.

3.4 DFAO Application and σ_T

In this section, we introduce an operation (or, depending on your perspective, a collection of operations) to our first-order theory, and extend Theorem 3.1 accordingly. The new operation allows us to apply a given DFAO $T\colon \Gamma^\omega \to \{0,1\}$ to arbitrary prefixes of an automatic sequence, although it is more general than that. To be precise, we introduce an operator σ_T for each DFAO $T\colon \Gamma^\omega \to \{0,1\}$. The input to σ_T is a collection of formulas $\{\phi_a\}_{a\in\Gamma}$ over a common set of d free variables, $\{x_1, \ldots, x_d\}$, with one distinguished free variable, x_i. Let $w \in \Gamma^{\omega^d}$ be the d-dimensional $\mathcal{N}$-automatic sequence corresponding to the formulas $\{\phi_a\}_{a\in\Gamma}$. Then $\sigma_T(\{\phi_a\}_{a\in\Gamma}, x_i)$ is a formula with free variables $x_1, \ldots, x_d$, such that

$$\sigma_T(\{\phi_a\}_{a\in\Gamma}, x_i)(a_1, \ldots, a_d)$$

is true if and only if $T(w_i[0..\langle a_i\rangle_{\mathcal{N}} - 1]) = 1$, where $w_i \in \Gamma^\omega$ is the $\mathcal{N}$-automatic sequence $n \mapsto w[a_1, \ldots, a_{i-1}, n, a_{i+1}, \ldots, a_d]$.

Since the definition of σ_T is somewhat confusing, let us give a couple of examples. Recall the Thue-Morse word, $\mathbf{t} \in \{0,1\}^\omega$ and $\mathbf{d} \in \{0,1\}^\omega$ from the previous chapter. The

period-doubling word is known to be the first-difference sequence of Thue-Morse word, modulo 2. That is,

$$\mathbf{d}[i] \equiv \mathbf{t}[i+1] - \mathbf{t}[i] \pmod 2$$

for all i. Hence, express $\mathbf{r}$ as a first-order predicate in terms of $\mathbf{t}$. Conversely, Thue-Morse is the running sum, modulo 2, of the period-doubling word.

$$\mathbf{t}[i] \equiv \sum_{j=0}^{i-1} \mathbf{d}[j] \pmod 2$$

Let $T\colon \{0,1\}^* \to \{0,1\}$ be an automaton that computes the sum of its inputs modulo 2 (coincidentally, T is the Thue-Morse automaton). Then $\sigma_T(\{\mathbf{t}[x] = a\}_{a\in\{0,1\}}, x)$ is a formula for $\mathbf{d}$. That is, $\sigma_T(\{\mathbf{t}[x] = a\}_{a\in\{0,1\}}, x)$ is true if and only if $\mathbf{d}[x] = 1$.

The formula is in the first-order theory $\mathsf{FO}(\mathbb{N}, <, +, P_0, P_1, \sigma_T)$, so once we extend Theorem 3.1 to that theory, we can construct an automaton for $\mathbf{t}$, given the automaton for $\mathbf{d}$.

Another example comes from an alternate definition of the Fibonacci word [6, Section 9.2]. Consider a line of slope $\phi = \frac{1+\sqrt{5}}{2}$, plotted on a square grid as pictured below in Figure 3.1. As we move along the line in the positive direction, it intersects the grid at vertical grid lines and horizontal grid lines. Now record the sequence of intersections, with 0 for intersections with horizontal grid lines and 1 for intersections with vertical grid lines. The resulting sequence is $\mathbf{f}$, the Fibonacci word.

Suppose that the line of slope ϕ reflects off of the first grid line it meets instead of passing through it. Since each grid line is an axis of symmetry for the entire grid, this does not change the sequence of intersections. If we reflect at each intersection, then the line bounces within a unit square. Let $\mathbf{g} \in \{\mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$ be the sequence of bounces, where the four sides are labelled `N`, `E`, `S`, and `W`, in clockwise order, starting from the top face.

$$\mathbf{g} = \mathtt{NESNWSENSWNSENWSNESWN}\cdots$$
$$\mathbf{f} = \mathtt{01001010010010100 1010}\cdots$$

Observe that $\mathbf{f}$ is $\mathbf{g}$ under a coding where $\mathtt{N}, \mathtt{S} \mapsto 0$, $\mathtt{E}, \mathtt{W} \mapsto 1$, because it does not distinguish between an `N` bounce and a `S` bounce, or likewise between `E` and `W`.

Clearly the `N`/`S` bounces alternate between `N` and `S` (starting with `N`), and similarly `E`/`W` bounces alternate between `E` and `W`. It is also clear that the functions $\psi_0\colon \{0,1\}^* \to \{N, S\}$

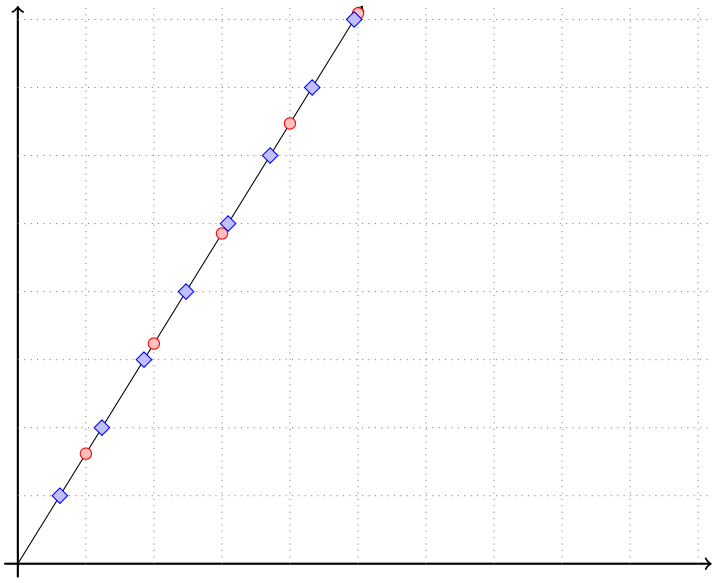


Figure 3.1: The Fibonacci word as a cutting sequence.

and $\psi_1 \colon \{0,1\}^* \to \{E, W\}$ given by

$$\psi_0(x) = \begin{cases} \mathtt{N}, & \text{if } |x|_0 \equiv 0 \pmod 2; \\ \mathtt{S}, & \text{if } |x|_0 \equiv 1 \pmod 2; \end{cases}$$
$$\psi_1(x) = \begin{cases} \mathtt{E}, & \text{if } |x|_1 \equiv 0 \pmod 2; \\ \mathtt{W}, & \text{if } |x|_1 \equiv 1 \pmod 2; \end{cases}$$

are automatic. If T_0 and T_1 are DFAOs for the automatic functions ψ_0 and ψ_1, then the sequences $(\psi_0(\mathbf{f}[0..n-1]))_{n=0}^{\infty}$ and $(\psi_1(\mathbf{f}[0..n-1]))_{n=0}^{\infty}$ are $\mathcal{F}$-automatic where $\mathcal{F}$ is the Fibonacci representation. Then we can recover $g[n]$ as

$$\mathbf{g}[n] = \begin{cases} \psi_0(\mathbf{f}[0..n-1]), & \text{if } \mathbf{f}[n] = 0; \\ \psi_1(\mathbf{f}[0..n-1]), & \text{if } \mathbf{f}[n] = 1; \end{cases}$$

for all n, so $\mathbf{g}$ is $\mathcal{F}$-automatic.

3.4.1 Implementation of σ_T

All our previous extensions of the first-order theory were based on predicates, but this operation is different. Our operation is what is known as a *variable-binding operator*. A familiar example of a variable-binding operator is existential quantification, $\exists x \phi(x)$, which binds a free variable x in some formula ϕ. Our operator is more like differentation, which performs a transformation on some variable x, but x remains free after the transformation. To add $\{\sigma_T\}_{T\colon \Gamma\to\{0,1\}}$ to our first-order theory and extend Theorem 3.1 accordingly, we need to prove that we can construct an ω-automaton for $\Lambda(\sigma_T(\{P_a\}_{a\in\Gamma}, x))$ given the DFAO T and ω-automata for $\{\Lambda(P_a)\}_{a\in\Gamma}$. Our goal is to prove Theorem 3.14, which shows that this construction is possible, extending our decidability results to

$$\mathsf{FO}(\mathbb{N}, <, +, V_k, \{\sigma_T\}_{T \text{ a DFAO}}, \{P_a\}_{a\in\Gamma})\,.$$

Then Corollary 3.15 extends our decidability results to $\mathcal{N}$-automatic sequences, for fairly general $\mathcal{N}$.

Consider the simplest case, where the predicates $\{P_a\}_{a\in\Gamma}$ describe a k-automatic sequence $w \in \Gamma^\omega$. If $\Lambda(\sigma_T(\{P_a\}_{a\in\Gamma}, x))$ is ω-regular (we will show that it is) then that means the sequence $(T(w[0..n-1]))_{n=0}^\infty$ is k-automatic. This special case was shown by Dekking in [25]. Barany uses a similar idea in [8]; he says (roughly translated to our terminology) an $\mathcal{N}$-automatic word $w \in \Gamma^\omega$ is *canonical* if for any monoid homomorphism $\psi\colon \Gamma^* \to M$ into a finite monoid M and $m \in M$, the set

$$\{(x, y) \in \mathbb{N} \times \mathbb{N} : x < y \wedge \psi(w[x..y]) = m\}$$

is automatic.

Recall that σ_T acts on formulas, which are permitted to be multidimensional. For instance, consider the predicate $P(i, j) := (\mathbf{t}[i + j] = 1)$, where $\mathbf{t}$ is the (2-automatic) Thue-Morse sequence. This formula describes the two-dimensional automatic sequence $i, j \mapsto \mathbf{t}[i + j]$. Fixing i gives us $\mathbf{t}[i..\infty]$, the Thue-Morse sequence shifted by i. Using Dekking [25], the sequence $(T(\mathbf{t}[i..i + j]))_{j=0}^\infty$ is 2-automatic for any fixed i. We will extend this to show that the 2-dimensional sequence $i, j \mapsto T(\mathbf{t}[i..i + j])$ is 2-automatic.

Definition 3.10. Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ be an LSD ideal numeration system. Suppose $f\colon (\Theta \times \Sigma)^* \to \Gamma$ is an automatic function. We define the *$\mathcal{N}$-linearization of f*, a function $\eta_f^{\mathcal{N}}\colon \Theta^* \to \Gamma^*$ such that $\eta_f^{\mathcal{N}}(\theta) = (f(\theta, x))_{x\in L\cap\Sigma^{|\theta|}}$, where x ranges over $L \cap \Sigma^{|\theta|}$ such that $\langle x\rangle_{\mathcal{N}}$ is monotonically increasing.

We write η_f for the linearization of f when $\mathcal{N}$ is understood. If the domain alphabet of f is of the form $\Psi \times \Sigma^d$ then there are d ways to partition it into $\Theta \times \Sigma$, one for each of

the Σ in the product. To distinguish these functions, we call the ith one the *linearization of f along the ith coordinate.*

For example, suppose $\mathcal{N}$ is the base-k numeration system and $w \in \Gamma^\omega$ is a k-automatic sequence. By definition, there is an automatic function $f\colon \Sigma_k^* \to \Gamma$ corresponding to w. If we let $\Theta = \{\square\}$ be a singleton set (recall our discussion of automata with zero inputs in Section 1.4.2), then we can define an automatic function $f'\colon (\Theta \times \Sigma_k)^* \to \Gamma$ where $f'(\theta, x) = f(x)$. Then η_f, the linearization of f, returns the prefix $w[0..k^n - 1]$ on input $\square^n \in \Theta^*$. Given a 2-dimensional automatic sequence, $w \in \Gamma^{\omega\times\omega}$, the linearization of f along the x-coordinate returns prefixes of the ith column on input $(i)_k \in \Sigma_k^*$, and similarly the linearization of f along the y-coordinate returns prefixes of the jth row on input $(j)_k \in \Sigma_k^*$.

Another example is the automatic function $f\colon \Sigma_k \times \Sigma_k \to \{\mathtt{a}, \mathtt{b}\}$ where

$$f(i,j) = \begin{cases} \mathtt{a}, & \text{if } i \neq j; \\ \mathtt{b}, & \text{if } i = j. \end{cases}$$

Since f is symmetric (i.e., $f(i,j) = f(j,i)$), linearizing f along either coordinate gives the same result, $\eta_f\colon \Sigma_k^* \to \{0,1\}$. Then $\eta_f(x)$ is a word of $\mathtt{a}$'s of length $k^{|x|}$, except for a single $\mathtt{b}$ at position $\langle x \rangle_k$, where $x \in \Sigma_k^*$.

We plan to address σ_T for the base-k numeration system first. Recall that for k-automatic sequences, Theorem 2.14 (Cobham's theorem) allows us to express a k-automatic sequence with k-uniform morphisms. We strive for the same kind of morphic characterization for linearizations of k-automatic functions.

Definition 3.11. Let $\mathrm{Unif}(\Sigma)$ denote the set of uniform morphisms from Σ^* to Σ^*. We note that the composition of two uniform morphisms is uniform, and composition with the *identity morphism* ($x \mapsto x$ for all $x \in \Sigma^*$) does nothing. Hence, $\mathrm{Unif}(\Sigma)$ is a monoid under composition of morphisms.

Lemma 3.12. *Let $k \geq 2$ be an integer and let $f\colon (\Theta \times \Sigma_k)^* \to \Gamma$ be a k-automatic function.*

Then there exists

- *a finite alphabet X,*
- *a symbol $c \in X$,*
- *a morphism $\Phi\colon \Theta^* \to \mathrm{Unif}(X)$ such that $\Phi(c)$ is k-uniform for all $c \in \Theta$, and*
- *a coding $\alpha\colon X^* \to \Gamma^*$*

such that

$$\eta_f(\theta) = \alpha(\Phi(\theta)(c))$$

for all $\theta \in \Theta^*$.

Note that we will write Φ_θ *for* $\Phi(\theta)$ *henceforth, because* $\Phi_\theta(c)$ *is easier to read than* $\Phi(\theta)(c)$.

Proof. Apply Theorem 1.14 to f. Then there exists a finite monoid M, a morphism $\mu\colon (\Theta \times \Sigma_k)^* \to M$ and a function $\tau\colon M \to \Gamma$ such that $f = \tau \circ \mu$.

Let $X = M$ be our finite alphabet, let c be the identity element in M and extend τ to a coding $\alpha\colon X^* \to \Gamma^*$. Define $\Phi\colon \Theta^* \to \mathrm{Unif}(X)$ so that

$$\Phi_\theta(x) = (\mu(\theta, 0) \cdot x)(\mu(\theta, 1) \cdot x) \cdots (\mu(\theta, k-1) \cdot x).$$

We claim that $\eta_\mu(\theta) = \Phi_\theta(c)$. Once we prove that, observe that $\eta_f(\theta) = \alpha \circ \eta_\mu(\theta)$ since

$$\eta_f(\theta)[\langle d \rangle_k] = f(\theta, d) = \alpha(\mu(\theta, d)) = \alpha(\eta_\mu(\theta)[\langle d \rangle_k]) = \alpha(\eta_\mu(\theta))[\langle d \rangle_k]$$

for all $\theta \in \Theta^*$ and $d \in \Sigma_k^{|\theta|}$. This completes the proof, because

$$\eta_f(\theta) = \alpha \circ \eta_\mu(\theta) = \alpha(\Phi_\theta(c)).$$

We now prove the claim by induction on $|\theta|$. If $|\theta| = 0$ then $\theta = \varepsilon$ and we have

$$\eta_\mu(\varepsilon) = \mu(\varepsilon, \varepsilon) = 1_M = c = \Phi_\varepsilon(c).$$

Otherwise, we let $\theta = \theta_1 \cdots \theta_n$. Now consider a word $d = d_1 \cdots d_n \in \Sigma_k^*$ and the corresponding position $\langle d_1 \cdots d_n \rangle_k$ in $\eta_\mu(\theta)$. By definition,

$$\begin{aligned}
\eta_\mu(\theta_1 \cdots \theta_n)[\langle d_1 \cdots d_n \rangle_k] &= \mu(\theta_1 \cdots \theta_n, d_1 \cdots d_n) \\
&= \mu(\theta_1, d_1) \cdot \mu(\theta_2 \cdots \theta_n, d_2 \cdots d_n) \\
&= \mu(\theta_1, d_1) \cdot \eta_\mu(\theta_2 \cdots \theta_n)[\langle d_2 \cdots d_n \rangle_k]
\end{aligned}$$

Recall that $\mu(\theta_1, d_1) \cdot m$ is the d_1'th symbol of $\Phi_{\theta_1}(m)$.

$$\begin{aligned}
\eta_\mu(\theta_1 \cdots \theta_n)[\langle d_1 \cdots d_n \rangle_k] &= \Phi_{\theta_1}\Big(\eta_\mu(\theta_2 \cdots \theta_n)[\langle d_2 \cdots d_n \rangle_k]\Big)[d_1] \\
&= \Phi_{\theta_1}\left(\eta_\mu(\theta_2 \cdots \theta_n)\right)\left[d_1 + k\,\langle d_2 \cdots d_n \rangle_k\right] \\
&= \Phi_{\theta_1}\left(\eta_\mu(\theta_2 \cdots \theta_n)\right)\left[\langle d_1 d_2 \cdots d_n \rangle_k\right]
\end{aligned}$$

We apply the induction hypothesis to $\eta_\mu(\theta_2 \cdots \theta_n)$ and obtain

$$\eta_\mu(\theta_1)[\langle d \rangle_k] = \Phi_\theta(c)[\langle d \rangle_k].$$

Since d was an arbitrary word in Σ_k^n, and $\Phi_\theta(c)$ has length k^n, it follows that $\eta_\mu(\theta) = \Phi_\theta(c)$, as desired. □

For example, the function $f(i,j) = \mathbf{t}[i+j]$, where $\mathbf{t}$ is the Thue-Morse word, is 2-automatic. Both linearizations of f are the same since $f(i,j) = f(j,i)$. We let η_f denote the linearization of f. Then $\eta_f(\theta)$ for $\theta \in \Sigma_2^*$ is a prefix of $\mathbf{t}[\langle \theta \rangle_2 ..\infty]$ (assuming LSD first). By the theorem, there exists

- an alphabet $X = \{0_0, 0_1, 1_0, 1_1\}$,
- a symbol $c = 0_1 \in X$,
- a morphism $\Phi\colon \Sigma_2^* \to \mathrm{Unif}(X)$ where

$$\begin{aligned} \Phi_0(0_0) &= 0_1 1_0 & \qquad \Phi_1(0_0) &= 1_0 0_1 \\ \Phi_0(0_1) &= 0_1 1_1 & \qquad \Phi_1(0_1) &= 1_1 1_0 \\ \Phi_0(1_0) &= 1_0 0_0 & \qquad \Phi_1(1_0) &= 0_0 0_1 \\ \Phi_0(1_1) &= 1_0 0_1 & \qquad \Phi_1(1_1) &= 0_1 1_0, \end{aligned}$$

- and a coding $\alpha\colon X^* \to \Sigma_2^*$ where

$$\begin{aligned} \alpha(0_0) &= 0 & \qquad \alpha(0_1) &= 0 \\ \alpha(1_0) &= 1 & \qquad \alpha(1_1) &= 1, \end{aligned}$$

such that $\eta_f(\theta) = \alpha(\Phi_\theta(c))$. Suppose we let $\theta = (n)_2 0^i$ be the LSD representation of a fixed $n \geq 0$ with i trailing zeros. Then $\langle \theta \rangle_2 = n$, so $\eta_f(\theta)$ is a prefix of $\mathbf{t}[n..\infty]$. On the other hand,

$$\begin{aligned} \eta_f(\theta) &= (\alpha \circ \Phi_{(n)_2})\Big(\Phi_{0^i}(0_1)\Big) \\ &= (\alpha \circ \Phi_{(n)_2})\Big(\Phi_0^i(0_1)\Big). \end{aligned}$$

It follows that $\mathbf{t}[n..\infty] = (\alpha \circ \Phi_{(n)_2})\Big(\Phi_0^\omega(0_1)\Big)$. That is, $\mathbf{t}[n..\infty]$ is the image of

$$\Phi_0^\omega(0_1) = 0_1 1_1 1_0 0_1 1_0 0_0 0_1 1_1 1_0 0_0 0_1 1_0 0_1 1_1 1_0 0_1 \cdots$$

under the uniform morphism $\alpha \circ \Phi_{(n)_2}$. Hence, any suffix of $\mathbf{t}$ (i.e., a shift) is the uniform morphic image of the word $\Phi_0^\omega(0_1)$.

As an interesting side note, consider a shift of $\mathbf{t}$ by -1. We have not defined $(-1)_2$, but under one natural definition (two's complement), the representation of -1 is the infinite word $1^\omega = 1111\cdots$. Hence, a shift by -1 should be $\Phi_0^\omega(0_1)$ under the uniform image of $\alpha \circ \Phi_{1^\omega}$. This is also subject to interpretation, but consider the images under α of the fixed points of Φ_{11}.

$$\begin{aligned}
\alpha(\Phi_{11}^\omega(0_0)) &= 0011010011001011010011001101\cdots \\
\alpha(\Phi_{11}^\omega(0_1)) &= 0100101100110100101101001100 10\cdots \\
\alpha(\Phi_{11}^\omega(1_0)) &= 1011010011001011010011001101\cdots \\
\alpha(\Phi_{11}^\omega(1_1)) &= 1100101100110100101101001100 10\cdots
\end{aligned}$$

These words are, respectively, $0\mathbf{t}$, $0\overline{\mathbf{t}}$, $1\mathbf{t}$, and $1\overline{\mathbf{t}}$, where $\mathbf{t}$ is the Thue-Morse word and $\bar{t}$ is the complement of the Thue-Morse word. In each case, we have prepended a single symbol to a fixed point of the Thue-Morse morphism; in other words, a shift by -1.

In the case of a k-automatic sequence w, Lemma 3.12 says that the prefix of length k^n is of the form $\alpha(\varphi^n(c))$. This is one direction of Theorem 2.14 (Cobham's theorem); the other half says that the fixed point of a uniform morphism φ under a coding α is a k-automatic sequence. Generalizing the second half of the theorem gives us the following lemma.

Lemma 3.13. *Fix an integer $k \geq 2$. Given*

- *a finite alphabet X,*
- *a symbol $c \in X$,*
- *a morphism $\Phi\colon \Theta^* \to \mathrm{Unif}(X)$ such that Φ_a is k-uniform for all $a \in \Theta$, and*
- *a coding $\alpha\colon X^* \to \Gamma^*$,*

there exists an automatic function $f\colon (\Theta \times \Sigma_k)^* \to \Gamma$ such that $\eta_f(\theta) = \alpha(\Phi_\theta(c))$, where η_f is the base-k linearization of f.

Proof. We want to show that the function $f\colon (\Theta \times \Sigma_k)^* \to \Gamma$ such that

$$f(\theta, d) := \alpha(\Phi_\theta(c))[\langle d \rangle_k],$$

is automatic. To do this, we find a monoid M, morphism $\mu\colon (\Theta \times \Sigma_k)^* \to M$ and map $h\colon M \to \Gamma$ such that $f = h \circ \mu$.

Then we define

- $M := X^X$, the set of functions from X to X under composition (i.e., $(f \cdot g)(x) = g(f(x))$),

- $\mu(\theta, d) := a \mapsto \Phi_\theta(a)[\langle d \rangle_k]$, and
- $h(g) := \alpha(g(c))$.

It is clear that

$$(h \circ \mu)(\theta, d) = \alpha(\Phi_\theta(c)[\langle d \rangle_k]) = f(\theta, d).$$

The proof is complete once we verify that μ is a morphism. Suppose $\theta_1, \theta_2 \in \Theta^*$ and $d_1, d_2 \in \Sigma_k^*$ such that $|\theta_1| = |d_1|$ and $|\theta_2| = |d_2|$. Then

$$\begin{aligned}\mu(\theta_1, d_1)\mu(\theta_2, d_2) &= (a \mapsto \Phi_{\theta_1}(a)[\langle d_1 \rangle_k])(a \mapsto \Phi_{\theta_2}(a)[\langle d_2 \rangle_k]) \\ &= a \mapsto \Phi_{\theta_1}\Big(\Phi_{\theta_2}(a)[\langle d_2 \rangle_k]\Big)[\langle d_1 \rangle_k]\end{aligned}$$

We observe that $\Phi_{\theta_1}(\Phi_{\theta_2}(a)[\langle d_2 \rangle_k])$ is

$$\Phi_{\theta_1\theta_2}(a)[k^{|\theta_1|} \langle d_2 \rangle_k \,..k^{|\theta_1|}(\langle d_2 \rangle_k + 1) - 1]$$

since Φ_{θ_1} is $k^{|\theta_1|}$-uniform. Then

$$\Big(\Phi_{\theta_1\theta_2}(a)[k^{|\theta_1|} \langle d_2 \rangle_k \,..k^{|\theta_1|}(\langle d_2 \rangle_k + 1) - 1]\Big)[\langle d_1 \rangle_k] = \Phi_{\theta_1\theta_2}(a)[\langle d_1 d_2 \rangle_k],$$

so we conclude that $\mu(\theta_1, d_1)\mu(\theta_2 d_2) = \mu(\theta_1\theta_2, d_1 d_2)$, and since $\mu(\varepsilon, \varepsilon)$ is the identity function, μ is a morphism. □

The following result says that we can apply an arbitrary automatic function g to prefixes of the linearization of an arbitrary automatic function f, and the result is another automatic function h. This is the main result of the chapter, since it shows that we can implement σ_T for the base-k numeration system, and a corollary generalizes to other numeration systems.

Theorem 3.14. *Fix an integer $k \geq 2$. Let $f\colon (\Theta \times \Sigma_k)^* \to \Gamma$ and $g\colon \Gamma^* \to \Delta$ be automatic functions. Define a function $h\colon (\Theta \times \Sigma_k)^* \to \Delta$ as follows. Let*

$$h(\theta, d) := g(\eta_f(\theta)[0..\langle d \rangle_k - 1])$$

where η_f is the base-k linearization of f. Then h is an automatic function.

Proof. We use Lemma 3.12 to construct a finite alphabet X, a symbol $c \in X$, a morphism $\Phi\colon \Theta^* \to \mathrm{Unif}(X)$ and a coding $\alpha\colon X^* \to \Gamma^*$ such that $\eta_f(\theta) = \alpha(\Phi_\theta(c))$.

Now apply Theorem 1.14 to g. We obtain a finite monoid M, a morphism $\mu\colon \Gamma^* \to M$, and a function $\tau\colon M \to \Delta$ such that $g = \tau \circ \mu$.

Using these decompositions, we can write h as the slightly messy expression

$$\begin{aligned} h(\theta, d) &= \tau(\mu(\alpha(\Phi_\theta(c))[0..\langle d\rangle_k - 1])) \\ &= \tau((\mu \circ \alpha)(\Phi_\theta(c)[0..\langle d\rangle_k - 1])). \end{aligned}$$

To clean up this expression, we compose $\mu\colon \Gamma^* \to M$ and $\alpha\colon X^* \to \Gamma^*$ into a single morphism $\beta\colon X^* \to M$. It suffices to show that $h'\colon (\Theta \times \Sigma_k)^* \to M$ such that $h'(\theta, d) = \beta(\Phi_\theta(c)[0..\langle d\rangle_k - 1])$ is a k-automatic function. Then $h = \alpha \circ h'$ is easily seen to be k-automatic.

To prove that h' is k-automatic, we find a finite monoid N, morphism $\rho\colon (\Theta \times \Sigma_k)^* \to N$ and map $\gamma\colon N \to M$ such that $h' = \gamma \circ \rho$. Then h' is k-automatic by Theorem 1.14.

Let M^X denote the set of morphisms from X^* to M. Note that M^X is finite because every morphism in M^X is determined by its restriction to X (see Theorem 1.5), and there are finitely many functions from X to M since M and X are finite. Let an arbitrary element of N consist of the following three components.

- A function $r\colon X \to X$. There are finitely many functions of this type since X is finite.
- A function $s\colon M^X \to M^X$ which maps morphisms in M^X to morphisms in M^X.
- A function $t\colon X \times M^X \to X$.

That is, N is based on the set $(X \to X) \times (M^X \to M^X) \times (X \times M^X \to X)$, and has the following monoid operation. The product of elements $(r_1, s_1, t_1) \in N$ and $(r_2, s_2, t_2) \in N$ is (r, s, t) where

$$\begin{aligned} r(\hat{c}) &= r_1(r_2(\hat{c})) \\ s(\hat{\beta}) &= s_2(s_1(\hat{\beta})) \\ t(\hat{c}, \hat{\beta}) &= t_2(\hat{c}, s_1(\hat{\beta})) \cdot t_1(r_2(\hat{c}), \hat{\beta}) \end{aligned}$$

for all $\hat{c} \in X$ and $\hat{\beta} \in M^X$.

We define $\rho(\theta, d) = (r, s, t)$ such that

$$\begin{aligned} r(\hat{c}) &= \Phi_\theta(\hat{c})[\langle d\rangle_k] \\ s(\hat{\beta}) &= \hat{\beta} \circ \Phi_\theta \\ t(\hat{c}, \hat{\beta}) &= \hat{\beta}(\Phi_\theta(\hat{c})[0..\langle d\rangle_k - 1]) \end{aligned}$$

for all $\theta \in \Theta^*$, $d \in \Sigma_k^*$, $\hat{c} \in X$ and $\hat{\beta} \in M^X$. We omit the proof that ρ is a morphism because it is tedious, and there is a similar proof in the previous theorem.

Then we conclude by defining $\gamma\colon N \to M$ so that $\gamma(r, s, t) = t(c, \beta)$, and therefore

$$\gamma(\rho(\theta, d)) = \beta(\Phi_\theta(c)[0..\langle d\rangle_k - 1]) = h'(\theta, d)$$

as desired. □

Note that Theorem 3.14 assumes base-k representation. We generalize the numeration system with the following corollary. This is one of the main contributions of this thesis.

Corollary 3.15. *Let $\mathcal{N} = (\Sigma, L, \langle\cdot\rangle_{\mathcal{N}})$ be an ideal numeration system with lexicographic comparison. Let $f\colon (\Theta \times \Sigma)^* \to \Gamma$ and $g\colon \Gamma^* \to \Delta$ be automatic functions. Define a function $h\colon (\Theta \times \Sigma)^* \to (\Delta \cup \{\varnothing\})$ as follows. Let*

$$h(\theta, d) := \begin{cases} g(\eta_f(\theta)[0..\langle d\rangle_{\mathcal{N}} - 1]), & \text{if } d \in L; \\ \varnothing, & \text{if } d \notin L; \end{cases}$$

where η_f is the $\mathcal{N}$-linearization of f. Then h is an automatic function.

Proof. The labels of elements Σ do not matter, except for the element $0 \in \Sigma$ implied by the fact that $\mathcal{N}$ is an ideal numeration system. Since 0 must be the minimum element in the order on Σ, we may assume without loss of generality that $\Sigma = \Sigma_k$ for some $k \geq 1$.

If $k = 1$ then there is at most one representation of any length, so $\eta_f(\theta)$ is either empty or a single character; we leave this trivial case to the reader, and proceed under the assumption that $k \geq 2$.

We define new functions, $f'\colon (\Theta \times \Sigma)^* \to (\Gamma \cup \{\varnothing\})$ and $g'\colon (\Gamma \cup \{\varnothing\})^* \to (\Delta \cup \{\varnothing\})$ based on f and g respectively. Let

$$f'(\theta, d) := \begin{cases} f(\theta, d), & \text{if } d \in L; \\ \varnothing, & \text{if } d \notin L; \end{cases}$$

and $g' = g \circ v$ where $v\colon (\Gamma \cup \{\varnothing\})^* \to \Gamma$ is a morphism such that $v(a) = a$ for all $a \in \Gamma$, but $v(\varnothing) = \varepsilon$.

Let $\eta_{f'}$ be the *base-k* linearization of f'. We observe that since base-k and $\mathcal{N}$ both have lexicographic comparison, all the symbols in $\eta_f(\theta)$ occur in $\eta_{f'}(\theta)$ in the same order, interspersed with runs of $\varnothing$. In other words, $v(\eta_{f'}(\theta)) = \eta_f(\theta)$.

We can feed f' and g' to Theorem 3.14 and obtain a function $h'\colon (\Theta \times \Sigma)^* \to \Delta$ where

$$\begin{aligned} h'(\theta, d) &= g'(\eta_{f'}(\theta)[0..\langle d\rangle_k - 1]) \\ &= g(v(\eta_{f'}(\theta)[0..\langle d\rangle_k - 1])). \end{aligned}$$

If $d \in L$ then we observe that $v(\eta_{f'}(\theta)[0..\langle d\rangle_k - 1])$ has length $\langle d\rangle_{\mathcal{N}}$, since precisely the representations for $0, 1, \ldots, \langle d\rangle_{\mathcal{N}} - 1$ are lexicographically less than d, and these representations correspond to the non-$\varnothing$ symbols in $\eta_{f'}(\theta)[0..\langle d\rangle_k - 1]$. Then

$$\begin{aligned} v(\eta_{f'}(\theta)[0..\langle d\rangle_k - 1]) &= v(\eta_{f'}(\theta))[0..\langle d\rangle_{\mathcal{N}} - 1] \\ &= \eta_f(\theta)[0..\langle d\rangle_{\mathcal{N}} - 1]. \end{aligned}$$

Clearly $h'(\theta, d) = g(\eta_f(\theta)[0..\langle d\rangle_{\mathcal{N}} - 1])$ for all $d \in L$, and h' is automatic. It follows that h is automatic. □

This is enough to extend Theorem 3.1 with the operation σ_T.

3.4.2 Applications of σ_T

Let us consider the applications and implications of Corollary 3.15. For the following examples, we assume $\mathcal{N}$ an ideal numeration system with lexicographic comparison.

- We recover Dekking's result [25] about k-automatic sequences. Let $w \in \Gamma^\omega$ be an $\mathcal{N}$-automatic sequence and let $T\colon \Gamma^* \to \Delta$ be a DFAO. Then $(T(w[0..n-1]))_{n=0}^\infty$ is an $\mathcal{N}$-automatic sequence. Dekking proves the special case where $\mathcal{N}$ is the base-k numeration system.

- We can apply a DFAO, $T\colon \Gamma^* \to \Delta$, to arbitrary subwords of an $\mathcal{N}$-automatic sequence $w \in \Gamma^\omega$. That is, the two-dimensional sequence $i, j \mapsto T(w[i..j-1])$ is $\mathcal{N}$-automatic. The trick is to construct a two-dimensional sequence $z \in \Gamma^{\omega\times\omega}$, where

$$z(i,j) := \begin{cases} w[j], & \text{if } i \le j; \\ \varnothing, & \text{if } j < i. \end{cases}$$

  Then create a DFAO $T'\colon (\Gamma \cup \{\varnothing\})^* \to \Delta$ that is identical to T except that the input alphabet includes $\varnothing$ and T' ignores any occurrence of $\varnothing$ in the input[2]. Applying the modified DFAO to z shows that the two-dimensional sequence

$$i, j \mapsto T'(z(i,0)z(i,1)\cdots\ z(i,j-1))$$

  is $\mathcal{N}$-automatic, and we observe that $T'(z(i,0)z(i,1)\cdots\ z(i,j-1)) = T(w[i..j-1])$.

[2]We used this technique in the proof of Corollary 3.15

- If we assume addition is automatic in the numeration system $\mathcal{N}$ then there is another way to apply a DFAO $T\colon \Gamma^* \to \Delta$ to subwords of an $\mathcal{N}$-automatic sequence $w \in \Gamma^\omega$. We construct the two-dimensional sequence $i, n \mapsto w[i+n]$. Then apply the DFAO T along variable j to show that $i, n \mapsto T(w[i..i+n-1])$ is $\mathcal{N}$-automatic. This is sometimes preferable to the sequence $i, j \mapsto T(w[i..j-1])$.

- Let $w \in \Gamma^\omega$ be an $\mathcal{N}$-automatic sequence where $\mathcal{N}$ is a numeration system with automatic addition. Given a DFAO $T\colon (\Gamma \times \Gamma)^* \to \Delta$, the three-dimensional sequence such that
$$i, j, n \mapsto T(w[i..i+n-1], w[j..j+n-1])$$
is $\mathcal{N}$-automatic. That is, we can apply T to any pair of subwords of the same length. This is a new result that allows us to compare two subwords using DFAOs. For instance, there is a DFAO $T_=\colon (\Gamma \times \Gamma)^* \to \{=, \neq\}$ that decides whether its two inputs are equal. We can show that $i, j, n \mapsto T_=(w[i..i+n-1], w[j..j+n-1])$ is $\mathcal{N}$-automatic, so we can compare two subwords for equality in our predicates. Of course, we already know how to compare subwords for equality (using the predicate $(\forall k < n\,(w[i+k] = w[j+k]))$), but the same principle applies to more complicated kinds of comparison. As an example, the *Hamming distance* $d(x, y)$ of two words $x, y \in \Sigma^n$ is the number of positions $1 \leq i \leq n$ such that $x[i] \neq y[i]$. We cannot compute the Hamming distance, but for any constant $m \geq 2$ there is a DFAO that computes the Hamming distance modulo m. We give another example of subword comparison below.
This easily generalizes from two subwords any fixed number of subwords. We note that Jason Bell gave an earlier, independent, unpublished proof of the same result in December 2012.
- Let $w \in \Sigma^\omega$ be an infinite word. The *abelian complexity* of w is the function $f\colon \mathbb{N} \to \mathbb{N}$ such that
$$f(n) := |\{\psi(x) : x \text{ a length-}n \text{ subword of } w\}|,$$
where $\psi\colon \mathbb{N} \to \mathbb{N}^\Sigma$ is the Parikh map. Naturally, a word has *bounded abelian complexity* if the abelian complexity function is bounded.
In a recent paper, Richomme, Saari and Zamboni give the following characterization of words with bounded abelian complexity.

Lemma 3.16. *Let $w \in \Sigma^*$ be a word, and let $C \geq 0$ be an integer. We say w is C-balanced if for any two factors x and y (of the same length) in w, we have $\|\psi(x) - \psi(y)\|_\infty \leq C$, where $\|(v_1, \ldots, v_n)\|_\infty := \max_i |v_i|$ is the infinity norm. In other words, the number of occurrences of any symbol $a \in \Sigma$ in x differs from the number of occurrences in y by at most C.*

Then w has bounded abelian complexity if and only if w is C-balanced for some C.

For instance, the Thue-Morse word and Fibonacci word are known to be 2-balanced and 1-balanced respectively, so they have bounded abelian complexity.
Given a constant C, we will show how to use DFAO-based subword comparison to test whether an automatic word $w \in \Sigma^*$ is C-balanced. A simple pumping argument shows there does not exist an automaton that decides whether $\|\psi(x) - \psi(y)\|_\infty \leq C$. On the other hand, there is an automaton M that accepts x and y if and only if

$$\|\psi(x[0..i-1]) - \psi(y[0..i-1])\|_\infty \leq C$$

for all i. The automaton M keeps track of the current vector $\psi(x[0..i-1]) - \psi(y[0..i-1])$ in $\{v \in \mathbb{N}^\Sigma : \|v\|_\infty \leq C\}$ and updates it as necessary when it reads the next pair of symbols $(x[i], y[i])$. If $\|\psi(x[0..i-1]) - \psi(y[0..i-1])\|$ exceeds C at any point, the automaton transitions to a "dead" non-accepting state, and stays there for the remainder of the input.
Clearly if M accepts on all pairs of subwords in w (of the same length), then w is C-balanced. Conversely, if w is C-balanced, then M accepts on all pairs of subwords in w (of the same length). We can use this technique to show that a word w is C-balanced (or not C-balanced) for a particular C.

Corollary 3.15 also has important applications related to monadic second-order formulas.

Theorem 3.17. *Let $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ be a formula where $x_1, \ldots, x_n$ are free variables over $\mathbb{N}$ and $X_1, \ldots, X_m$ are free variables over subsets of $\mathbb{N}$. Let $\mathcal{N}$ be an ideal numeration system with lexicographic comparison. Given $\mathcal{N}$-automatic sets $A_1, \ldots, A_m \subseteq \mathbb{N}$, the language $\Lambda(\phi(x_1, \ldots, x_n, A_1, \ldots, A_m))$ is ω-regular.*

Proof. The language $\Lambda_\infty(\phi)$ is ω-regular by Theorem 3.7, so it is recognized by some Muller automaton $M = (\{0,1\}^{m+n}, Q, \delta, q_0, \mathbf{F})$. Let $M' = (\{0,1\}^{m+n}, Q, \delta, q_0, Q, \gamma)$ be a DFAO with the same set of states, initial state, and transitions as M, and such that the output of state q is $\gamma(q) := q$.

The words $\chi(A_1), \ldots, \chi(A_m)$ are $\mathcal{N}$-automatic, since the corresponding sets $A_1, \ldots, A_m$ are $\mathcal{N}$-automatic. Similarly, $\chi(i)$ is $\mathcal{N}$-automatic for any fixed $i \in \mathbb{N}$, and the two-dimensional sequence $i, j \mapsto \chi(i)[j]$ is $\mathcal{N}$-automatic. Hence,

$$f(x_1, \ldots, x_n, i) := (\chi(x_1)[i], \ldots, \chi(x_n)[i], \chi(A_1)[i], \ldots, \chi(A_m)[i])$$

is an $n+1$-dimensional $\mathcal{N}$-automatic sequence.

We use $\sigma_{M'}$ along variable i of the automatic sequence f. This gives us an automaton for the $\mathcal{N}$-automatic sequence

$$f'(x_1, \ldots, x_n, i) := M'\Big((\chi(x_1), \ldots, \chi(x_n), \chi(A_1), \ldots, \chi(A_m))[0..i-1]\Big).$$

Recall the acceptance condition for a Muller automaton: M accepts a word w if it the subset of states $S \subseteq Q$ encountered infinitely often (while reading w) is in $\mathbf{F}$. We can set this up as a first-order predicate,

$$(\exists^\infty i\, f'(x_1, \ldots, x_n, i) = q).$$

It follows that the set

$$\{(a_1, \ldots, a_n) \in \mathbb{N}^n : \phi(a_1, \ldots, a_n, A_1, \ldots, A_n)\}$$

is $\mathcal{N}$-automatic, and therefore $\Lambda(\phi(x_1, \ldots, x_n, A_1, \ldots, A_m))$ is ω-regular. □

The proof tells us something about the relative strength of the monadic second-order theory $\mathsf{MSO}(\mathbb{N}, <, P_1, \ldots, P_m)$, and $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma}, \{\sigma_T\}_{T \text{ a DFAO}})$. Note that in the proof of Theorem 3.17, we turn a given formula in $\mathsf{MSO}(\mathbb{N}, <, P_1, \ldots, P_m)$ (assuming $P_1, \ldots, P_m$ are the predicates associated with $A_1, \ldots, A_m$) into a DFAO M', and then turn that into a formula in $\mathsf{FO}(\mathbb{N}, <, P_1, \ldots, P_m, \sigma_{M'})$. Hence, any query we can state in $\mathsf{MSO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ can be translated into a query in $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma}, \{\sigma_T\}_{T \text{ a DFAO}})$.

Conversely, we can express σ_T in $\mathsf{MSO}(\mathbb{N}, <)$. Suppose we want to compute an automaton for the predicate $\sigma_T(\{\phi_a\}_{a\in\Gamma}, y)$, where $T = (\Gamma, Q, \delta, q_0, F)$ is an arbitrary DFA (which we interpret as a DFAO to $\{0, 1\}$), and $\{\phi_a\}_{a\in\Gamma}$ are predicates in $\mathsf{MSO}(\mathbb{N}, <)$. For simplicity, we assume the ϕ_a's describe a sequence $w \in \Gamma^*$. Let us consider the subsets $\{X_q\}_{q\in Q}$ of $\mathbb{N}$ where

$$X_q = \{n \in \mathbb{N} : \delta^*(q_0, w[0..n-1]) = q\}.$$

That is, X_q contains integers n such that reading n symbols of w with T leaves us in state q. Observe that we can also define X_q by the following properties:

- $0 \in X_{q_0}$,
- $n \in X_q \implies n + 1 \in X_{\delta(q,w[n])}$ for all n, and
- for all n, there is a unique $q \in Q$ such that $n \in X_q$.

These properties are expressible in $\mathsf{MSO}(\mathbb{N}, <)$, since they use only set membership, successor, existential and universal quantifiers and binary connectives. Then $T(w[0..n-1]) = 1$ if and only if n is in X_q for some $q \in F$. We conclude that σ_T we can express σ_T in $\mathsf{MSO}(\mathbb{N}, <)$.

It appears that $\mathsf{FO}(\mathbb{N}, <, \{\sigma_T\}_{T \text{ a DFAO}})$ and $\mathsf{MSO}(\mathbb{N}, <)$ can express the same first-order (i.e., having no free set variables) predicates on $\mathbb{N}$. It is intruiging that when we augment these theories with addition, $\mathsf{MSO}(\mathbb{N}, <, +)$ is undecidable, but $\mathsf{FO}(\mathbb{N}, <, \{\sigma_T\}_{T \text{ a DFAO}}, +)$ is decidable.

Chapter 4

Applications

We present applications and extensions of the results in Chapter 3. In Section 4.1, we discuss a series of problems in which we think of a k-automatic set $S \subseteq \mathbb{N} \times \mathbb{N}$ as defining a set of rational numbers, $\{\frac{a}{b} : (a, b) \in S\} \subseteq \mathbb{Q}$, and show how to decide certain properties of these sets. In Section 4.2, we extend the decidability results from the previous chapter to paperfolding words, an uncountably infinite collection of binary words.

4.1 Critical Exponent and k-Automatic Sets of Rational Numbers

This section is based on work in [49], but improves on it with a more efficient algorithm for one of the main decidability results. Computing the critical exponent is an excellent application of this algorithm. Let us start by defining powers and fractional powers in infinite words.

Definition 4.1. Let $x = x[1..n] \in \Sigma^*$ be a finite word. We say x *has period* p or p *is a period of* x if $x[i] = x[i + p]$ for all $1 \leq i \leq n - p$. We say p *is the period of* x if p is the least period of x.

If x is a word with period p and length q such that $p < q$ then we say x is a *fractional power* with *exponent* $\frac{q}{p}$, or *a* $\frac{q}{p}$*-power*.

Much work has been done on avoiding fractional powers in infinite words. In particular, Dejean's conjecture was recently confirmed by Currie and Rampersad [21], and independently by Rao [43].

Theorem 4.2 (Dejean's Conjecture)**.** *Define the* repetition threshold over an n symbol alphabet*,* $\mathrm{RT}(n)$*, to be*

$$\mathrm{RT}(n) := \inf\{d \in \mathbb{Q} : \textit{an infinite word over } n \textit{ symbols avoids } d\textit{-powers}\}.$$

Then $\mathrm{RT}(n)$ *takes the following values for* $n \geq 2$*.*

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

For instance, $\mathrm{RT}(6) = \frac{6}{5}$, so we can avoid k-powers for $d > \frac{6}{5}$ on an alphabet of size 6, but we cannot avoid d-powers for $d < \frac{6}{5}$.

A related question is, which d-powers does a specific infinite word avoid, and what is the threshold? This is known as the *critical exponent* of the word, defined below.

Definition 4.3. Let $w \in \Sigma^\omega$ be an infinite word. Then the *critical exponent* of w, denoted by $\mathrm{ce}(w)$, is defined to be

$$\mathrm{ce}(w) := \sup\{d \in \mathbb{Q} : w \text{ contains a } d\text{-power subword}\}.$$

Allouche, Rampersad, and Shallit [4] noticed that we can express "$w \in \Sigma^*$ contains a word of length q with period p" as a predicate in $\mathsf{FO}(\mathbb{N}, <, +, \{P_a\}_{a\in\Sigma})$ as follows

$$(\exists i \ (\forall j \ (j + p \geq q) \vee (w[i + j] = w[i + j + p]))).$$

Hence, if w is a k-automatic sequence then the set

$$S = \{(p, n) \in \mathbb{N}^2 : w \text{ contains a word of length } n \text{ with period } p\}$$

is k-automatic. We would like to interpret these pairs as fractions, because then the critical exponent is

$$\mathrm{ce}(w) = \sup\left\{\frac{n}{p} : (p, n) \in S\right\}.$$

We have reduced computing the critical exponent of a k-automatic sequence to finding the supremum of the rational numbers accepted by an automaton.

This situation is not unique to critical exponent – there are many examples of k-automatic sets $S \subseteq \mathbb{N}^2$ which we think of as defining a set of rational numbers, $\{\frac{a}{b} : (a, b) \in S\} \subseteq \mathbb{Q}$. This idea is explored by Rowland and Shallit [46], and subsequently by Schaeffer and Shallit [49] in the context of critical exponent.

Definition 4.4. Let $S \subseteq \mathbb{Q}$ be a set of rational numbers. We say S *is k-automatic* if there exists a k-automatic set $T \subseteq \mathbb{N}^2$ such that

$$S = \left\{ \frac{a}{b} : (a, b) \in T \right\}.$$

We list a number of examples, mostly taken from [49], which illustrate practical uses for k-automatic rational sets.

- Critical exponent with predicates. We can consider the critical exponent for a subset of the subwords in w. That is,

  $$\mathrm{ce}_P(w) := \sup\{\exp(x) : x \text{ is a subword of } w \text{ and } P(x) \text{ is true}\}$$

  where P is a predicate on words. As long as we can express P in our first order theory, the set

  $$\{(n, p) \in \mathbb{N}^2 : w \text{ contains a subword with period } p \text{ and length } n \text{ satisfying } P\}$$

  is k-automatic, and hence $\mathrm{ce}_P(w)$ is the supremum of a k-automatic set of rational numbers. Some interesting examples of predicates are
  - x is a prefix of w,
  - x occurs infinitely often, and
  - $|x|$ is larger than some constant c.

- Critical exponent with finitely many exceptions. Recall the Rudin-Shapiro sequence,

  $$\mathbf{r} = 000100100001110100010010111000\cdots$$

  from Chapter 1. The critical exponent of $\mathbf{r}$ is 4 because it contains the subword 0000. Since 0000 and 1111 are the *only* 4-powers in $\mathbf{r}$, one could argue that the critical exponent misrepresents the threshold between avoidable and unavoidable exponents in $\mathbf{r}$. We can use predicates to ignore 0000 and 1111, then the next highest exponent in Rudin-Shapiro is 3, associated with the subwords 000 or 111. If we omit those, then the next highest exponent in Rudin-Shapiro is $\frac{8}{3}$, corresponding to subwords 00100100 and 11011011, and so on. It turns out that if we omit all factors of length 11 or less (00010000100 is a factor of length 11 with exponent $\frac{11}{5}$) then all remaining subwords have exponent at most 2. Since there are arbitrarily large subwords with exponent 2, we cannot lower the exponent any further without omitting an infinite set of subwords.

In some cases, we can achieve a descending sequence of critical exponents by omitting larger and larger finite sets. For instance, consider the word $\mathbf{x}$ obtained from the Thue-Morse word, $\mathbf{t}$, by flipping bits at positions $4^k - 1$ for all $k \geq 0$. That is,

$$\mathbf{x} = 11111001100101111001011001101001\cdots$$

Then the flipped bit $4^k - 1$ induces an overlap $\mathbf{x}[4^k - 1..3 \cdot 4^k - 1]$ of length $2 \cdot 4^k + 1$ and period 4^k for all $k \geq 0$. The exponents of these subwords decrease to 2, and all other subwords (except the prefix 11111 and its subwords) in $\mathbf{x}$ have exponent at most 2. In this situation, it makes sense to define

$$\inf\{\alpha \in \mathbb{R} : \alpha \leq \exp(x) \text{ for finitely many words } x \text{ in } w\}.$$

This is essentially lim sup of the k-automatic rational set

$$\{(n, p) \in \mathbb{N}^2 : w \text{ contains a subword with period } p \text{ and length } n\}.$$

Technically, it is the largest *special point* (defined later in Section 4.1.2) of the set, but special points coincide with limit points in this example.
For many of the examples in this list, we can replace sup with "largest special point" and obtain a new, interesting example.

- Linear recurrence. A word $w \in \Sigma^\omega$ is *linearly recurrent* if there exists a constant C such that for all $n \geq 1$, every window of length $\lfloor Cn \rfloor$ contains all subwords of length n in w. We can use k-automatic rational sets to find the optimal constant C^* for a linearly recurrent k-automatic sequence.
  Given n and ℓ, the predicate

$$P(n, \ell) = (\forall i \ (\forall j \ w[i..i + \ell - 1] \text{ contains } w[j..j + n - 1]))$$

tests if every window of length ℓ contains every subword of length n. Assuming that w is recurrent (i.e., for all n there exists ℓ_n such that $P(n, \ell)$), then the predicate

$$Q(n, \ell) = P(n, \ell) \wedge (\forall \ell_0 < \ell \ \neg P(n, \ell))$$

is true only when ℓ is the minimal window length. Since Q is expressible as a predicate, the set

$$S = \{(n, \ell) \in \mathbb{N}^2 : Q(n, \ell)\}$$

is k-automatic. Then C is greater than $\frac{\ell}{n}$ for all $(n, \ell) \in S$, and the optimal constant C^* is the least such upper bound, so

$$C^* = \sup\left\{\frac{\ell}{n} : (n, \ell) \in S\right\},$$

the supremum of a k-automatic set of rational numbers.
Similarly, we can obtain optimal constants bounding the ratio of appearance window length to subword length, and condensation window length to subword length.

With these examples in mind, let us develop the theory of k-automatic rational sets.

4.1.1 Basic Operations on k-Automatic Rational Sets

We discuss some basic operations on subsets of $\mathbb{N}^2$ that apply to k-automatic rational sets.

Proposition 4.5. *Let $X \subseteq \mathbb{N}^2$ be a k-automatic set. Then for constants $a, b, c, d, e, f \in \mathbb{N}$, the set*

$$Y = \{(ax + by + c, dx + ey + f) : (x, y) \in X\} \subseteq \mathbb{N}^2$$

is also k-automatic.

This can be interpreted as a Möbius transformation of the rational set. The result still holds, to a limited extent, when the constants a, b, c, d, e, f are integers.

Another problem is detecting "invalid" rational numbers, such as $\frac{0}{0}$ or $\frac{1}{0}$. Fortunately, we can efficiently find and remove these numbers from the set.

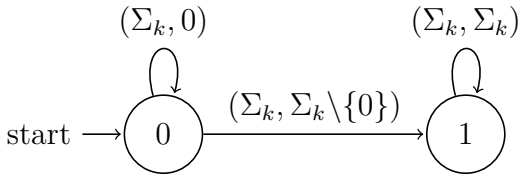
Proposition 4.6. *Let $X \subseteq \mathbb{N}^2$ be a k-automatic set. Given a DFA T for X (either MSD first or LSD first), we can decide whether X contains elements of the form $(a, 0)$ in linear time. We can also construct an automaton for the set*

$$\{(a, b) \in X : b \neq 0\}$$

in linear time, with at most twice as many states.

Proof. We can detect if T accepts any strings of the form $\{(a, 0) : a \in \Sigma_k\}^*$ by deleting all other transitions and checking if any final state is reachable. This takes linear time.

To construct an automaton for $\{(a, b) \in X : b \neq 0\}$, we take the product of T with the following DFA.



□

In the same vein, we can efficiently detect whether the rational set is unbounded.

Proposition 4.7. *Let $X \subseteq \mathbb{N}^2$ be a k-automatic set. Given a DFA T for X (either MSD first or LSD first), we can decide, in linear time, whether $\{\frac{a}{b} : (a,b) \in X\}$ is unbounded.*

Proof. Let us consider the MSD-first and LSD-first automata separately.

Suppose T reads numbers starting from the most significant digit. If $\{\frac{a}{b} : (a,b) \in X\}$ is unbounded then T accepts words where b has an arbitrarily large number of leading zeros (and a has no leading zeros). By a pumping argument,

- there is a cycle C in T with zeros in the second coordinate,
- there is a path from the initial state to the cycle, with zeros in the second coordinate and at least one nonzero symbol in the first coordinate, and
- there is a path from the cycle to a final state.

We can test T for these properties in linear time. Conversely, if T contains such a cycle then $\{\frac{a}{b} : (a,b) \in X\}$ is clearly unbounded.

Suppose T reads numbers starting from the least significant digit. Then similarly, $\{\frac{a}{b} : (a,b) \in X\}$ is unbounded if and only if T accepts words where b has arbitrarily many trailing zeros. This is equivalent to

- a cycle C in T with zeros in the second coordinate,
- a path from the initial state to the cycle, and
- a path from the cycle to some final state, with zeros in the second coordinate and at least one nonzero symbol in the first coordinate.

□

This is a useful first step when we compute the supremum, to rule out the possibility that it is infinite.

4.1.2 Limit Points and Special Points

Consider a k-automatic rational set

$$S = \left\{\frac{a}{b} : (a,b) \in X\right\} \subseteq \mathbb{Q}$$

and recall the definition of a limit point.

Definition 4.8. Let $S \subseteq \mathbb{R}$ be a set of real numbers. A *limit point of* S is a real number $\alpha \in \mathbb{R}$ such that for any neighbourhood $U \subseteq \mathbb{R}$ of α, the set $U \cap S$ is infinite.

A somewhat surprising example is the k-automatic set $X = \{(a, b) \in \mathbb{N}^2 : a = 4b\}$. Counterintuitively, the corresponding set of rationals is just $\{4\}$ and hence has no limit points, despite the fact that X contains infinitely many representations of 4. It turns out to be easier, and arguably more natural, to treat numbers with infinitely many representations like limit points.

Definition 4.9. Let $X \subseteq \mathbb{N}^2$ be a k-automatic set. Then a *special point of* X is a real number $\alpha \in \mathbb{R}$ such that for any neighbourhood $U \subseteq \mathbb{R}$ of α, the set

$$\left\{(a, b) \in X : \frac{a}{b} \in U\right\}$$

is infinite.

Proposition 4.10. *Let $X \subseteq \mathbb{N}^2$ be a k-automatic set and let $S := \{\frac{a}{b} : (a, b) \in X\}$. If $\alpha \in \mathbb{R}$ is a limit point of S then α is a special point of X.*

The converse does not hold, since 4 is a special point of $\{(a, b) \in \mathbb{N}^2 : a = 4b\}$, but the corresponding rational set $S = \{4\}$ has no limit points.

4.1.3 Computing the Supremum and Largest Special Point

In this section we will show how to compute the supremum/infimum and largest/smallest special point of a k-automatic rational set using linear programming. We give a series of lemmas, relating the points and limit points to infinite walks in a graph, relating those infinite walks to solutions of an optimization problem, and then characterizing the optimal solution to the problem. These lemmas culminate in Theorem 4.20, which shows that there is a polynomial-time algorithm to compute the supremum of a k-automatic rational set. This improves on the algorithm in [49].

For convenience, we let $X \subseteq \mathbb{N}^2$ be a k-automatic set, let $S = \{\frac{a}{b} : (a, b) \in X\}$ be the corresponding set of rationals. We need to work closely with the actual base-k representations, so let $L \subseteq (\Sigma_k \times \Sigma_k)^*$ be the (regular) language

$$L := \{((a, b))_k : (a, b) \in X\}$$

with most significant digit first.

Let $\pi_1, \pi_2 \colon (\Sigma_k \times \Sigma_k)^* \to \Sigma_k^*$ denote codings that projects to a word to its first or second component respectively. Define a function $\mathrm{quo}_k \colon (\Sigma_k \times \Sigma_k)^* \to \mathbb{R}$ that maps $x = x[1..n]$ to the fraction

$$\frac{\langle \pi_1(x) \rangle_k}{\langle \pi_2(x) \rangle_k}.$$

This is equivalent to

$$\frac{\sum_i \pi_1(x[i]) k^{-i}}{\sum_i \pi_2(x[i]) k^{-i}}$$

which naturally extends to an infinite word $x = x[1..\infty] \in (\Sigma_k \times \Sigma_k)^\omega$. Note that the infinite summations in the numerator and denominator converge because all terms are nonnegative and $\pi_j(a)k^{-i}$ is bounded by k^{-i+1}.

Lemma 4.11. *Suppose $\alpha \in \mathbb{R}$ is a real number. Then α is a special point of X if and only if there exists an infinite word $x \in (\Sigma_k \times \Sigma_k)^\omega$ such that $\mathrm{quo}_k(x) = \alpha$ and every prefix of x is a prefix of some word in L.*

Proof. Suppose α is a special point of X. Then there exists an infinite sequence of distinct representations $\{x_i\}_{i=0}^\infty$ such that $x_i \in L$ for all i and $\lim_{i\to\infty} \mathrm{quo}_k(x_i) = \alpha$. We may take a strictly increasing or strictly decreasing subsequence, so assume without loss of generality that the x_is are monotonic.

Infinitely many of the words in $\{x_i : i \geq 0\}$ begin with some symbol $c_0 \in \Sigma_k \times \Sigma_k$. Of those beginning with c_0, infinitely many begin with c_0c_1, and so on. In this fashion, we construct an infinite word $\mathbf{c} = c_0c_1c_2\cdots$, such that every prefix of $\mathbf{c}$ is a prefix of infinitely many words in $\{x_i : i \geq 0\} \subseteq L$. If x_j and $\mathbf{c}$ agree on a prefix of length n, then

$$\left(\sum_i \pi_\ell(x_j[i])k^{-i}\right) - \left(\sum_{i=0}^{\infty} \pi_\ell(\mathbf{c}[i])k^{-i}\right) = O(k^{-n})$$

for $\ell = 1, 2$. Hence, there is a subsequence $x_{i_1}, x_{i_2}, \cdots$ of $\{x_i\}_{i=0}^\infty$ such that

$$\lim_{n\to\infty} \sum_j \pi_\ell(x_{i_n}[j])k^{-j} = \sum_{i=0}^{\infty} \pi_\ell(\mathbf{c}[j])k^{-j}$$

and hence

$$\begin{aligned}\alpha &= \lim_{n\to\infty} \mathrm{quo}_k(x_{i_n}) \\ &= \frac{\lim_{n\to\infty} \sum_j \pi_1(x_{i_n}[j])k^{-i_n}}{\lim_{n\to\infty} \sum_j \pi_2(x_{i_n}[j])k^{-i_n}} \\ &= \frac{\sum_{i=0}^{\infty} \pi_1(\mathbf{c}[j])k^{-j}}{\sum_{i=0}^{\infty} \pi_2(\mathbf{c}[j])k^{-j}} \\ &= \mathrm{quo}_k(\mathbf{c}).\end{aligned}$$

Note that division by zero is not a problem, since $\pi_2(\mathbf{c}) = 0^\omega$ if and only if there are x_is with an arbitrarily large number of zeros in the denominator, and hence $\mathrm{quo}_k(x_i)$ grows arbitrarily large. Therefore we can say $\mathrm{quo}_k(\mathbf{c}) = \infty$ if and only if $\lim_{i\to\infty} \mathrm{quo}_k(x_i) = \infty$.

In the other direction, suppose $\alpha = \mathrm{quo}_k(\mathbf{c})$ where $\mathbf{c} \in (\Sigma_k \times \Sigma_k)^*$ is an infinite word such that every prefix of $\mathbf{c}$ is a prefix of some word in L. Let $x_i \in L$ be the shortest word that begins with the prefix $\mathbf{c}[0..i-1]$ for all $i \geq 0$. It is not hard to show that $\alpha = \mathrm{quo}_k(\mathbf{c}) = \lim_{i\to\infty} x_i$. Hence, for any neighbourhood of α, there is some x_i such that $\mathrm{quo}_k(x_i)$ is in the neighbourhood. Therefore α is a special point. □

Lemma 4.12. *There exists a directed graph $G = (V, E)$ with a distinguished initial vertex $v_0 \in V$ and edges labelled by symbols $\{\ell_e\}_{e\in E} \subseteq \Sigma_k \times \Sigma_k$ such that $\alpha \in \mathbb{R}$ is a point of S or special point of X if and only if $\alpha = \mathrm{quo}_k(x)$ where $x \in (\Sigma_k \times \Sigma_k)^\omega$ is the sequence of labels along some infinite walk in G.*

Proof. Let $T = (\Sigma_k \times \Sigma_k, Q, \delta, q_0, F)$ be a minimal DFA for the language L. We define the directed graph $G = (Q \cup \{q_F\}, E)$ so that it is the underlying directed graph of T (including labels, and potentially parallel edges) with the following modifications:

- an additional state/vertex, q_F,
- a loop from q_F to itself labelled $(0, 0) \in \Sigma_k \times \Sigma_k$, and
- a directed edge, labelled $(0, 0)$, from each final state $q \in F$ to q_F.

We will assume that there is a path from any state to q_F. If not, then there is some state in T, sometimes known as the *dead state*, which cannot reach any final state. There is at most one such state by minimality (otherwise we could merge them), and we simply delete it along with all transitions into it.

Let α be a point in S, so $\alpha = \mathrm{quo}_k(w)$ for some word $w \in L$. There is a finite walk in T from the initial state, q_0, to some final state, labelled by w. From the final state, we follow

the transition $(0,0)$ to q_F and then loop forever on $(0,0)$. This infinite walk, $w(0,0)^\omega$ has quotient

$$\mathrm{quo}_k(w(0,0)^\omega) = \mathrm{quo}_k(w) = \alpha,$$

as required. Similarly, if α is a special point then there is an infinite word $x \in (\Sigma_k \times \Sigma_k)^*$ such that $\alpha = \mathrm{quo}_k(x)$ and every prefix of x can be extended to a word in L. But then $\delta^*(q_0, x[0..n]) \in Q$ for all $n \geq 0$, so x is an infinite path in G through the states in Q, so α is of the form $\mathrm{quo}_k(x)$.

In the other direction, suppose $x \in (\Sigma_k \times \Sigma_k)^*$ is an infinite walk in G. Either the walk eventually reaches state q_F and loops forever, in which case we construct a finite word in L based on the prefix up to q_F and $\mathrm{quo}_k(x)$ is a point in S, or the walk meanders through Q forever, in which case every prefix can be extended to a word in L, so $\mathrm{quo}_k(x)$ is a special point of S. □

The supremum of S is either a point in S, or a limit point of S and therefore a special point of X. Hence, $\sup S$ is of the form $\mathrm{quo}_k(x)$, where $x \in (\Sigma_k \times \Sigma_k)^\omega$ labels an infinite walk in G. Our next step is to define an optimization problem such that an infinite walk in G labelled $x \in (\Sigma_k \times \Sigma_k)^\omega$ induces a feasible solution with objective value $\mathrm{quo}_k(x)$.

Definition 4.13. A *linear-fractional program* is an optimization problem of the form

$$\begin{aligned} &\max_{\mathbf{x}} \frac{\mathbf{c}^T\mathbf{x} + \alpha}{\mathbf{d}^T\mathbf{x} + \beta} \\ &\text{subject to} \\ &\quad A\mathbf{x} \leq \mathbf{b} \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c}, \mathbf{d} \in \mathbb{R}^n$, $\alpha, \beta \in \mathbb{R}$ are given constants and $\mathbf{x} \in \mathbb{R}^n$ is a vector of variables.

We assume that the denominator $\mathbf{d}^T\mathbf{x} + \beta$ is positive for all feasible solutions, because of the problems associated with division by zero.

Linear-fractional programming generalizes linear programming by letting the objective function be a quotient of two linear functions. This is clearly useful if we want the objective value to be $\mathrm{quo}_k(x)$. However, the following result of Charnes and Cooper [16] shows that we can convert a linear-fractional program to a linear program without changing the range of feasible objective values.

Theorem 4.14 (Charnes-Cooper transformation)**.** *Let (P) be the feasible, bounded linear-fractional program shown below.*

$$\begin{aligned} &\max_{\mathbf{x}} \frac{\mathbf{c}^T\mathbf{x}+\alpha}{\mathbf{d}^T\mathbf{x}+\beta} \\ &\textit{subject to} \\ &A\mathbf{x} \leq \mathbf{b} \end{aligned}$$

Then by introducing variables $\mathbf{y} = \frac{1}{\mathbf{d}^T\mathbf{x}+\beta}\mathbf{x} \in \mathbb{R}^n$ and $t = \frac{1}{\mathbf{d}^T\mathbf{x}+\beta} \in \mathbb{R}$, we may transform (P) into the following equivalent linear program, (P'):

$$\begin{aligned} &\max_{\mathbf{y},t} \mathbf{c}^T\mathbf{y}+\alpha \\ &\textit{subject to} \\ &A\mathbf{y} \leq \mathbf{b}t \\ &\mathbf{d}^T\mathbf{y}+\beta t = 1 \\ &t \geq 0. \end{aligned}$$

Proof. See [16]. □

Next, we define a linear-fractional program based on walks in G, so that a walk labelled $w \in (\Sigma_k \times \Sigma_k)^*$ corresponds to a feasible solution $\mathbf{x}(w)$ with objective value $\mathrm{quo}_k(\mathbf{x}(w))$.

Definition 4.15. Let (P) be a linear-fractional program with non-negative variables $\{x_e\}_{e\in E}$, linear constraints

$$\begin{aligned} \frac{1}{k}\sum_{e\in \mathrm{in}(v)} x_e &= \sum_{e\in \mathrm{out}(v)} x_e, && \text{for all } v \in V\backslash\{v_0\}, \\ 1+\frac{1}{k}\sum_{e\in \mathrm{in}(v_0)} x_e &= \sum_{e\in \mathrm{out}(v_0)} x_e, \end{aligned}$$

and objective function

$$\frac{\sum_{e\in E} x_e\pi_1(\ell_e)}{\sum_{e\in E} x_e\pi_2(\ell_e)}.$$

We seek to maximize the objective function.

Theorem 4.16. *Suppose there is an infinite walk $u = u[0..\infty] \in E^\omega$ in G starting from v_0. Then we have a feasible solution $\mathbf{x}(u) = \{x_e\}_{e\in E}$ where*

$$x_e = \sum_{i:u[i]=e} k^{-i}.$$

for all $e \in E$. Furthermore, the objective value of this solution is $\mathrm{quo}_k(w)$, where $w \in (\Sigma_k \times \Sigma_k)^\omega$ is the sequence of labels corresponding to the walk u.

Proof. First, $\mathbf{x}(u)$ is a feasible solution because $u[i]$ is an edge into v if and only if $u[i+1]$ is an edge out of v. Hence,

$$\begin{aligned}
\frac{1}{k}\sum_{e\in\mathrm{in}(v)} x_e &= \frac{1}{k}\sum_{e\in\mathrm{in}(v)}\sum_{i:u[i]=e} k^{-i} \\
&= \sum_{i:u[i]\in\mathrm{in}(v)} k^{-i-1} \\
&= \sum_{i:u[i+1]\in\mathrm{out}(v)} k^{-i-1} \\
&= \sum_{i>1:u[i]\in\mathrm{out}(v)} k^{-i} \\
&= \sum_{e\in\mathrm{in}(v)}\sum_{i:u[i]=e} k^{-i} \\
&= \sum_{e\in\mathrm{out}(v)} x_e
\end{aligned}$$

for $v \neq v_0$. The case $v = v_0$ is similar, but we miss the $k^0 = 1$ term for $u[0]$ and must add it to both sides.

The objective value of $\mathbf{x}(u)$ is

$$\begin{aligned}
\frac{\sum_{e\in E} x_e\pi_1(\ell_e)}{\sum_{e\in E} x_e\pi_2(\ell_e)} &= \frac{\sum_{e\in E}\sum_{i:u[i]=e}\pi_1(\ell_e)k^{-i}}{\sum_{e\in E}\sum_{i:u[i]=e}\pi_2(\ell_e)k^{-i}} \\
&= \frac{\sum_i \pi_1(\ell_{u[i]})k^{-i}}{\sum_i \pi_2(\ell_{u[i]})k^{-i}} \\
&= \frac{\sum_i \pi_1(w[i])k^{-i}}{\sum_i \pi_2(w[i])k^{-i}} \\
&= \mathrm{quo}_k(w),
\end{aligned}$$

as desired. □

The feasible region of (P) is a convex polytope (as a consequence of basic linear programming theory), so any convex combination of feasible solutions is again a feasible solution.

Theorem 4.17. *Suppose $\mathbf{x} = \{x_e\}_{e \in E}$ is a basic feasible solution to (P). Then $\mathbf{x}$ is of the form $\mathbf{x}(u)$ for $u \in E^\omega$ an ultimately periodic infinite walk in G starting at v_0. The optimal objective value of $\mathbf{x}$ is also rational.*

Proof. Note that $\mathbf{x}$ is a basic solution, so it is defined as the solution to a system of $|E|$ non-degenerate linear equations, $\mathcal{S}$. Each equation in $\mathcal{S}$ is either $x_e = 0$ or one of the linear constraints in (P).

Call x_e the *weight* of an edge e. Let U be the set of vertices such that some outgoing edge has nonzero weight. Hence, there are at least $|U|$ edges of nonzero weight.

On the other hand, if v is a vertex not in U then all its outgoing edges have weight zero. Since $\mathbf{x}$ is a feasible solution, all incoming edges also have weight zero. We may assume without loss of generality that the constraint in (P) corresponding to vertex v is not in $\mathcal{S}$, since otherwise we can replace it with $\{x_e = 0 : \text{for all edges } e \text{ incident to } v\}$. Hence, at most $|U|$ equations in $\mathcal{S}$ are taken from (P). It follows that at most $|U|$ edges have nonzero weight, since $\mathcal{S}$ contains at least $|E| - |U|$ equations of the form $x_e = 0$.

We see that there must be exactly $|U|$ edges of nonzero weight, one leaving each vertex in U. To construct our infinite walk $u \in E^\omega$, we start at v_0 (note that $v_0 \in U$) and follow the unique outgoing edge of nonzero weight for each step. Clearly u is ultimately periodic, and begins to repeat after visiting each vertex in U.

The corresponding feasible solution $\mathbf{x}(u)$ satisfies all the linear constraints in (P), and $\mathbf{x}(u)_e = 0$ for all edges e with weight zero in $\mathbf{x}$. Therefore $\mathbf{x}(u)$ satisfies all the equations in $\mathcal{S}$, so $\mathbf{x} = \mathbf{x}(u)$ because $\mathbf{x}$ is defined to be the solution to $\mathcal{S}$. Then the objective value of $\mathbf{x}(u)$ is $\mathrm{quo}_k(u)$. Since u is ultimately periodic, it is of the form $u = xy^\omega$ where $x, y \in (\Sigma_k \times \Sigma_k)^*$. Then

$$\begin{aligned}
\mathrm{quo}_k(xy^\omega) &= \frac{\langle \pi_1(x) \rangle_k + \sum_{i=1}^{\infty} \langle \pi_1(y) \rangle_k \, k^{-|y|i}}{\langle \pi_2(x) \rangle_k + \sum_{i=1}^{\infty} \langle \pi_2(y) \rangle_k \, k^{-|y|i}} \\
&= \frac{\langle \pi_1(x) \rangle_k + \langle \pi_1(y) \rangle_k \, \frac{k^{-|y|}}{1-k^{-|y|}}}{\langle \pi_2(x) \rangle_k + \langle \pi_2(y) \rangle_k \, \frac{k^{-|y|}}{1-k^{-|y|}}} \\
&= \frac{(k^{|y|}-1)\langle \pi_1(x) \rangle_k + \langle \pi_1(y) \rangle_k}{(k^{|y|}-1)\langle \pi_2(x) \rangle_k + \langle \pi_2(y) \rangle_k} \in \mathbb{Q},
\end{aligned}$$

so the objective value is rational. □

We state a classical result known as the *mediant inequality*, which we use to prove a corollary of Theorem 4.17.

Proposition 4.18. *Let $a, b, c, d > 0$ be real numbers such that $\frac{a}{b} \leq \frac{c}{d}$. Then*

$$\frac{a}{b} \leq \frac{a+b}{c+d} \leq \frac{c}{d}.$$

Corollary 4.19. *There is an optimal basic solution for (P).*

Proof. Every feasible solution $\mathbf{x}$ is a convex combination of a finite set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of basic feasible solutions. Say

$$\mathbf{x} = \alpha_1 \mathbf{x}_1 + \cdots + \alpha_n \mathbf{x}_n$$

for $\alpha_1, \ldots, \alpha_n > 0$ such that $\sum_i \alpha_i = 1$. Suppose that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are in non-decreasing order by objective value. By the mediant inequality,

$$\frac{\mathbf{c}^T(\alpha_1 \mathbf{x}_1 + \cdots + \alpha_{n-1}\mathbf{x}_{n-1})}{\mathbf{d}^T(\alpha_1 \mathbf{x}_1 + \cdots + \alpha_{n-1}\mathbf{x}_{n-1})} \leq \frac{\mathbf{c}^T \mathbf{x}}{\mathbf{d}^T \mathbf{x}} \leq \frac{\mathbf{c}^T \alpha_n \mathbf{x}_n}{\mathbf{d}^T \alpha_n \mathbf{x}_n}.$$

Therefore $\mathbf{x}_n$ is an optimal basic solution to (P). □

Theorem 4.20. *The* sup, inf, *largest special point, and smallest special point of a k-automatic rational set S are rational, and computable in polynomial time, given a DFA for the language $L \subseteq (\Sigma_k \times \Sigma_k)^*$.*

Proof. Given the DFA, we can check that the denominator is never zero (see Proposition 4.6) and detect if S is unbounded (see Proposition 4.7) in linear time, and act accordingly.

Let us consider the supremum first. The supremum is either a point of S, or a limit point of S and hence a special point in X. By Lemma 4.12, the supremum of S is of the form $\mathrm{quo}_k(x)$ where $x \in (\Sigma_k \times \Sigma_k)^\omega$ labels an infinite walk in G starting at v_0.

We define a linear-fractional program (P). Theorem 4.16 shows that for any walk in G (starting from v_0) labelled w, the objective value $\mathrm{quo}_k(x)$ is feasible. The optimal value for (P) is therefore at least $\sup S$, since it is at least as large as any point or special point.

On the other hand, every basic solution corresponds to an infinite walk in G (and hence a point or special point). If (P) were a linear program, this would be enough to conclude that there is an optimal basic solution. Since (P) is a linear-fractional program, we are

more cautious and prove this fact in Corollary 4.19. The optimal value of (P) is a point of S or special point of X, so it is bounded above by $\sup S$.

It follows that the optimal value of (P) is $\sup S$. We can convert (P) to a linear program using the Charnes-Cooper transformation (see Theorem 4.14), and then solve the program in polynomial time by an interior point method. Hence, we can compute $\sup S$ in polynomial time.

We can compute $\inf S$ with the same program (P), except we minimize the objective value instead of maximizing it. For the largest special point, observe that in Lemma 4.12, the walks corresponding to special points are precisely the walks that avoid the additional vertex $q_F \in V$. If we delete this vertex, or add a constraint that set the incoming or outgoing edge weights to zero, then infinite walks in G just correspond to special points in X. Then the optimal value of the program is the largest special point. Similarly for the smallest special point, by minimizing instead of optimizing. □

As a corollary, the critical exponent of a k-automatic sequence is rational and computable. Similarly, all of the examples we saw earlier are rational and computable, including the variants of critical exponent and linear recurrence constant. Our polynomial-time algorithm improves on the result in [49], but unfortunately it does not yield an efficient algorithm for critical exponent because the algorithm implied by Theorem 3.1 is not guaranteed to run in polynomial time or produce a DFA with polynomially many states. However, the DFA is usually not large or expensive to construct, so Theorem 4.20 is useful in practice.

4.2 Decidability for Paperfolding Words

The *regular paperfolding word* is an infinite binary word $\mathbf{p} = \mathbf{p}[1..\infty] \in \Sigma_2^\omega$. One definition of the paperfolding word is based on an iterated folding process. We fold a long, thin strip of paper in half repeatedly and then completely unfold it. This leaves a binary sequence of creases on the paper, since each crease turns either up or down. Suppose that in the middle of unfolding, the sequence of creases along the strip is $w \in \Sigma_2^*$. When we unfold one more time, we have creases w on the initial half of the strip, followed by a crease in the middle, and then the sequence w in reverse and upside down. Hence, unfolding maps the sequence w to $w1\overline{w}^R$, where $\overline{x}$ denotes the complement of x and x^R denotes the reversal of x. Iterating this map yields an infinite sequence of words with a limit, $\mathbf{p}$, the paperfolding word.

$$\mathbf{p} = 110110011100100\cdots$$

The *paperfolding words* are a family of infinite binary words, which include the *regular paperfolding word* and generalize it. Observe that every time we fold the strip of paper, we can choose to fold 'up' or 'down'. When we unfold, one choice leads to the map $f_1(w) := w1\overline{w}^R$, and the other choice leads to $f_0(w) \mapsto w0\overline{w}^R$. Suppose we have an infinite binary *instruction sequence* $\mathbf{a} \in \{0,1\}^\omega$ where $\mathbf{a}[n] \in \{0,1\}$ indicates whether the nth time we unfold is 'up' or 'down'. Hence, the sequence of creases after n unfolds is w_n where

$$\begin{aligned} w_0 &= \varepsilon \\ w_n &= f_{\mathbf{a}[n]}(w_{n-1}) \qquad\qquad \text{for all } n \geq 1. \end{aligned}$$

Define a word, $\mathbf{p_a} := \lim_{n\to\infty} w_n$, as the limit of these words. We call $\mathbf{p_a}$ the *paperfolding word associated with the unfolding instruction sequence* $\mathbf{a}$. The regular paperfolding word is the paperfolding word associated with the constant instruction sequence $1111\cdots$.

Proposition 4.21. *Let $\mathbf{a} \in \{0,1\}^\omega$ be an instruction sequence, and let $\mathbf{p_a}$ be the corresponding paperfolding word. Let $n \geq 1$ be an integer of the form $n = q \cdot 2^e$ for q odd. Then*

$$\mathbf{p_a}[n] := \begin{cases} \mathbf{a}[e], & \text{if } q \equiv 1 \pmod 4; \\ 1 - \mathbf{a}[e], & \text{if } q \equiv 3 \pmod 4. \end{cases}$$

Proof. See Theorem 6.5.2 in [6, p. 182]. □

As a corollary, we get a DFAO that defines all paperfolding sequences.

Corollary 4.22. *There exists a DFAO, F, (shown in Figure 4.1) which, given $\mathbf{a}[1..n]$ and a binary word $x \in \Sigma^*$ of length $|w| = n$, outputs $\mathbf{p_a}[\langle x \rangle_2]$.*

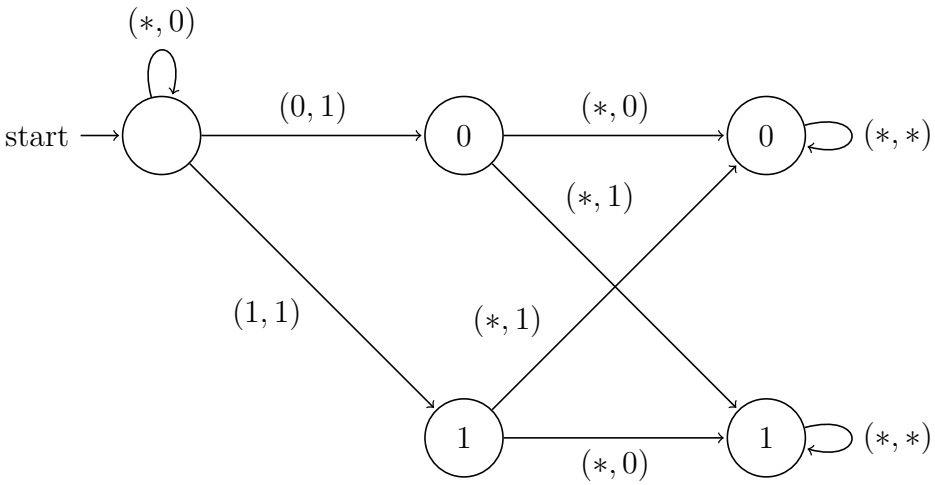


Figure 4.1: DFAO for all paperfolding sequences.

It follows that the regular paperfolding word, corresponding to $\mathbf{a} = 111\cdots$, is 2-automatic. We simply delete transitions where the instruction is 0, and drop the instruction component from all remaining transitions. Similarly, whenever $\mathbf{a}$ is ultimately periodic, we can manipulate the DFAO in Figure 4.1 to show that $\mathbf{p_a}$ is 2-automatic. On the other hand, there are only countably many automata and uncountably many paperfolding words, so most paperfolding words are not 2-automatic.

Theorem 4.23. *Let* $\mathbf{a}$ *be a sequence of unfolding instructions, and let* $\mathbf{p_a}$ *be the corresponding paperfolding word. Then* $\mathbf{p_a}$ *is* 2*-automatic if and only if* $\mathbf{a}$ *is ultimately periodic.*

Proof. See Theorem 6.5.4 in [6, p. 183]. □

In other words, our decidability results from the previous chapter do not apply to most paperfolding words because most paperfolding words are not 2-automatic. We will use F to extend the first-order theory so that we can decide questions about *all* paperfolding words.

4.2.1 First-Order Theory for Paperfolding Words

For a 2-automatic paperfolding word $\mathbf{q}$, we decide predicates in $\mathsf{FO}(\mathbb{N}, <, +, P_0, P_1)$ with Theorem 3.1. Recall that in the proof of Theorem 3.1, we use the fact that $\mathbf{q}$ is 2-automatic

to construct ω-automata for $\Lambda(P_0)$ and $\Lambda(P_1)$. Now we want to generalize the first-order theory and decidability results to the family of paperfolding words, and we know that the automaton F should play some role.

We propose to use a two-sorted first-order theory $\mathsf{FO}(\mathbb{N}, \Sigma_2^\omega, <, +, V_2)$, in which every variable has a *sort*: it is either a natural number (in $\mathbb{N}$) or an instruction sequence (in Σ_2^ω). In other words, we augment $\mathsf{FO}(\mathbb{N}, <, +, V_2)$ with instruction sequence variables (over the domain Σ_2^ω), and we include quantification ($\exists$, $\forall$) and comparison ($=$, $<$) over instruction sequences. Then F suggests a predicate F on instruction sequences and natural numbers, where $F(\mathbf{a}, n)$ is true if and only if $\mathbf{p_a}[n] = 1$. We include F in our first-order theory of paperfolding words, which we denote $\mathsf{FO}(\mathbb{N}, \Sigma_2^\omega, <, +, V_2, F)$.

Theorem 4.24. *Let $L_\omega \subseteq \Sigma_2^\omega$ denote the set of all binary representations of natural numbers.*

Given any logical formula $\phi(\mathbf{a}_1, \ldots, \mathbf{a}_m, i_1, \ldots, i_n)$ expressible in $\mathsf{FO}(\mathbb{N}, \Sigma_2^\omega, <, +, V_2, F)$, define a language of infinite words $\Lambda(\phi) \subseteq (\Sigma_2^{m+n})^\omega$ where

$$\Lambda(\phi) = \{(\mathbf{a}_1, \ldots, \mathbf{a}_m, i_1, \ldots, i_n) \in (\Sigma_2^\omega)^m \times L_\omega^n : \phi(\mathbf{a}_1, \ldots, \mathbf{a}_m, \langle i_1 \rangle_2, \ldots, \langle i_n \rangle_2)\}.$$

Then $\Lambda(\phi)$ is ω-regular, and we can effectively construct the corresponding ω-automaton.

Proof. Essentially all the work has already been done, most of it in Theorem 3.1.

- The constructions in Theorem 3.1 for $\vee$, $\wedge$ and $\neg$ work without modification.
- Quantification over natural numbers is unchanged, and quantification over instruction sequences. In a nutshell, use a coding to remove the quantified variable and determinize.
- Natural number equality and inequality comparison are handled by the same automata as in Theorem 3.1.
- V_2 and addition of binary numbers are covered in Section 3.2.1. These operations are unchanged.
- There exist simple automata to compare instructions sequences lexicographically, or for equality.

The only genuinely new element is the predicate F, and it is clear that $\Lambda(F)$ is ω-regular, because we can construct an ω-automaton for $\Lambda(F)$ from F. This completes the proof. □

Let us give some examples of queries we can state with $\mathsf{FO}(\mathbb{N}, \Sigma_2^\omega, <, +, V_2, F)$ and decide using Theorem 4.24. Many of these examples are due to Shallit, and taken from an unpublished draft.

- Narad Rampersad made the following unpublished conjecture in 2006, which we can now verify mechanically.

  Theorem 4.25. *Suppose* $\mathbf{a}, \mathbf{b} \in \Sigma_2^*$ *are distinct instruction sequences. Then there exists an integer* n *such that the set of length* n *subwords in* $\mathbf{p_a}$ *is disjoint from the set of length* n *subwords in* $\mathbf{p_b}$.
  In fact, if i *is the first position where* $\mathbf{a}$ *and* $\mathbf{b}$ *differ, then the sets of length* k *subwords in* $\mathbf{p}_a$ *and* $\mathbf{p}_b$ *are disjoint if and only if* $k \geq 14 \cdot 2^i$.

  Proof. We can express the claims as predicates. The first claim can be expressed as

  $$(\exists n\,(\forall i, j\,(\mathbf{p}_a[i..i+n-1] \neq \mathbf{p}_b[j..j+n-1])))$$

  where $\mathbf{p}_a[i..i+n-1] \neq \mathbf{p}_b[j..j+n-1]$ is short for

  $$(\exists k\,((k < n) \wedge (F(\mathbf{a}, i+k) \neq F(\mathbf{b}, i+k)))).$$

  For the second claim, observe that we can express "n is a power of 2" as $V_2(n) = n$. Since $\mathbf{p}_a[2^i] = \mathbf{a}[i]$, the predicate

  $$(\mathbf{p}_a[n] = \mathbf{p}_b[n]) \wedge (V_2(n) = n)$$

  is true if $n = 2^i$ where $\mathbf{a}$ and $\mathbf{b}$ differ at position i. We leave the rest of the predicate as an exercise to the reader. □

- We can determine the set of squares, cubes, and higher powers in paperfolding words. See the work of Allouche and Bousquet-Mélou [1].

  Theorem 4.26. *Let* $\mathbf{p}_f$ *be any paperfolding word. Then we make the following claims about powers in* $\mathbf{p}_f$
  - *Any square in* $\mathbf{p}_f$ *has length* 2, 6 *or* 10.
  - *The only cubes in* $\mathbf{p}_f$ *are* 000 *and* 111.
  - *There are no fractional dth-powers in* $\mathbf{p}_f$ *for* $d \geq 3$.

  Proof. A subword $\mathbf{p}_f[i..i+2n-1]$ is a square if and only if $\mathbf{p}_f[i..i+n-1] = \mathbf{p}_f[i+n..i+2n-1]$. Therefore we can state the first claim as

  $$\begin{aligned}&(\forall f \in \{0,1\}^\omega\\ &\quad(\forall n \geq 1\\ &\qquad(\exists i \geq 1\,\mathbf{p}_f[i..i+n-1] = \mathbf{p}_f[i+n..i+2n-1]) \iff n \in \{1,3,5\})).\end{aligned}$$

The second claim is similar. For higher powers, note that a subword $\mathbf{p}_f[i..i+n-1]$ has period r if

$$\mathbf{p}_f[i..i+n-r-1] = \mathbf{p}_f[i+r..i+n-1].$$

Then we can verify with predicates that for all f, for all subwords $\mathbf{p}_f[i..i+n-1]$, the minimum period r satisfies $3r \geq n$ (so the exponent, $\frac{n}{r}$, is at most 3). □

- Every paperfolding word $\mathbf{p}_f$ has a corresponding appearance function, $A_f\colon \mathbb{N} \to \mathbb{N}$. Recall that $A_f(n)$ is the minimum length such that $\mathbf{p}_f[1..A_f(n)]$ contains all subwords of length n in $\mathbf{p}_f$. We define a function $A\colon \mathbb{N} \to \mathbb{N}$ that is the maximum of these appearance functions. That is,

$$A(n) := \max_{f \in \{0,1\}^\omega} A_f(n)$$

for all n. It is not necessarily clear that the maximum exists, since $f \in \{0,1\}^*$ is unbounded, but we can prove it mechanically with the predicate

$$(\forall n(\exists c(\forall f \in \{0,1\}^\omega\, (A_f(n) \leq c))))$$

Furthermore, we can show that

$$\begin{aligned} &A(1) = 3, \\ &A(2) = 7, \\ &A(n) = 6 \cdot 2^i + n - 1, \qquad\qquad \text{where } 2^{i-1} < n \leq 2^i. \end{aligned}$$

In fact, $A(n) = A_f(n)$ where $f = 0001^\omega$. Since $\mathbf{p}_{0001^\omega}$ is 2-automatic by Theorem 4.23, we can compute the appearance function as we did in Chapter 3. Then we can mechanically decide the predicate

$$\begin{aligned} &(\forall f \in \{0,1\}^\omega \\ &\quad (\forall n \geq 1 \\ &\quad\quad (\forall i \geq 1 \\ &\quad\quad\quad (\exists j \leq A(n) - n \\ &\quad\quad\quad\quad \mathbf{p}_f[i..i+n-1] = \mathbf{p}_f[j..j+n-1])))) \end{aligned}$$

to verify that $A(n) \geq A_f(n)$ for all f and n.
Similarly, we can compute the minimum appearance function and the recurrence function.

- We can prove the following theorem about subword complexity, due to Allouche [2].

  Theorem 4.27. *For all $f \in \{0,1\}^\omega$, the subword complexity of $\mathbf{p}_f$ is $\rho\colon \mathbb{N} \to \mathbb{N}$ where*

$$\begin{aligned}
\rho(1) &= 2 \\
\rho(2) &= 4 \\
\rho(3) &= 8 \\
\rho(4) &= 12 \\
\rho(5) &= 18 \\
\rho(6) &= 23 \\
\rho(n) &= 4n, \qquad \text{for all } n \geq 7.
\end{aligned}$$

  For ordinary k-automatic sequences, it is non-trivial to express subword complexity as a predicate [30]. Fortunately, since the first difference sequence $(\rho(n)-\rho(n-1))_{n=1}^{\infty}$ is k-automatic (in fact, ultimately constant), we can verify the conjecture as follows.

  1. Verify the conjecture by hand for $n \leq 7$. The maximum appearance function, $A(n)$, indicates that we only need to consider the first $A(7) = 54$ symbols.

  2. We say a subword x is *right special* if $x0$ and $x1$ are both subwords. If x is not right special, then either $x0$ or $x1$ is a subword, since any occurrence of x is followed by some symbol. Let $s(n)$ be the number of right special subwords of length n. Observe that the $s(n)$ right special subwords of length n give us $2s(n)$ subwords of length $n+1$, and the remaining $\rho(n)-s(n)$ length-n subwords yield $\rho(n) - s(n)$ subwords of length $n+1$. Therefore

$$\rho(n+1) = \rho(n) + s(n),$$

     so it suffices to show that $s(n) = 4$ for all $n \geq 8$.

- Observe that Theorem 3.14 applies to paperfolding words. Suppose ϕ is a first-order formula with corresponding automatic function $f\colon (\Theta \times \Sigma_k)^* \to \{0,1\}$ corresponding to $\Lambda(\phi)$. When we apply the theorem, we linearize along one of the integer free variables and group all other free variables (integers and instructions) into Θ. One application involves the generalized Rudin-Shapiro sequences, defined below.

  Definition 4.28. Let $f \in \{0,1\}^\omega$ be an infinite sequence of unfolding instructions. Define the associated *generalized Rudin-Shapiro word*, $\mathbf{r}_f \in \{0,1\}^\omega$, such that

$$\mathbf{r}_f[i] = \sum_{j=0}^{i-1} \mathbf{p}_f \bmod 2.$$

There is clearly an automaton T over $\{0,1\}$ that computes the sum of its input bits modulo 2 (in fact, this is the Thue-Morse automaton). Using Theorem 3.14, we can transform the automaton for the family of paperfolding words into an automaton for the family of generalized Rudin-Shapiro sequences.

- Consider the following lemma about ω-regular languages, which we present without proof.

  Lemma 4.29. *Suppose Σ is an ordered finite alphabet. Let $L \subseteq \Sigma^\omega$ be a non-empty ω-regular language. Then the lexicographically least (or similarly, greatest) element of L is ultimately periodic.*

  Recall that by Theorem 4.24, any formula ϕ in our first-order theory corresponds to an ω-regular language $\Lambda(\phi)$. Applying the lemma gives us the following interesting corollary.

  Corollary 4.30. *Let ϕ be any formula in the first-order theory of paperfolding words. Either ϕ is unsatisfiable (any assignment of values to free variables renders ϕ false) or there is a satisfying assignment where all instruction sequence variables are ultimately periodic sequences (and hence, by Theorem 4.23, correspond to 2-automatic paperfolding words).*

  Conversely, given an ultimately periodic sequence $f \in \{0,1\}^\omega$, we can use the fact that $\mathbf{p}_f[2^k] = f[k]$ to construct a formula ϕ that is satisfied only by f. If f has period p and preperiod n then

$$\left(\bigwedge_{j=0}^{n+p-1} \mathbf{p}_g[2^j] = f[j] \right)$$

  checks that $g[0..n+p-1]$ matches $f[0..n+p-1]$, and

$$(i > 2^n) \vee (i = V_2(i)) \vee (\mathbf{p}_g[i] = \mathbf{p}_g[2^p i])$$

  ensures that $g[n..\infty]$ is periodic with period p. Combining these two formulas gives us a formula ϕ, such that f is the only satisfying assignment for the free variable g.

4.2.2 Paperfolding with DFAs and DFAOs

Recall that for a single automatic sequence $w \in \Gamma$, we can mechanically evaluate first order predicates using DFAs or DFAOs instead of ω-automata. The difficulty with DFAs and DFAOs is that there are multiple representations of each natural number. In particular,

we perform existential quantification by deleting the quantified variable and determinizing the remaining automaton, but if all satisfying assignments for that variable have longer representations than the rest of the variables, the resulting automaton may be incorrect. For ordinary automatic sequences, we devised an operation (see Theorem 3.6) to repair the automaton, allowing us to use DFAs.

For paperfolding sequences, the situation is complicated by the instructions, which we receive in parallel with the rest of the input. Consider, as a concrete example, the predicate "there exists x of the form $2^k - 1$ such that $\mathbf{p}_f[x] = \mathbf{p}_f[y]$". Given x, y and f as input, there is an automaton that decides whether x is of the form $2^k - 1$ and $\mathbf{p}_f[x] = \mathbf{p}_f[y]$. Now recall that $\mathbf{p}_f[2^k - 1] = f[k]$, so the predicate is equivalent to "there exists $f[k]$ such that $f[k] = \mathbf{p}_f[y]$". We cannot say that there does not exist k such that $f[k] = \mathbf{p}_f[y]$ without knowing all of f, so a DFA that takes a finite prefix of f as input will be unable decide certain inputs. Hence, we have been unable to extend Theorem 3.6 for formulas involving paperfolding sequences.

Despite the theoretical difficulties described above, we can use DFAs to decide a number of practical queries about paperfolding sequences, but we must state the queries much more carefully. The key observation is that there is a linear universal upper bound for the recurrence function of paperfolding words.

Theorem 4.31. *There exists a function $R\colon \mathbb{N} \to \mathbb{N}$ such that*

- *$R(n)$ is in $O(n)$, and*
- *for any paperfolding word $\mathbf{p}_f$, the corresponding recurrence function $R_f\colon \mathbb{N} \to \mathbb{N}$ has the property that $R_f(n) \leq R(n)$ for all n.*

Proof. See Allouche and Bousquet-Mélou [3]. They show that $R(n) = 44n$ is an upper bound for the recurrence function. Note that we can also verify this statement with a full implementation (based on ω-automata) of the algorithm implied by Theorem 4.24. □

How does Theorem 4.31 help us? Consider the predicate "there exists a palindrome of length n in $\mathbf{p}_f$". A priori, the length-n palindrome could be anywhere in $\mathbf{p}_f$, forcing us to check every $\mathbf{p}_f[i..i+n-1]$, and hence requiring knowledge of the entire instruction sequence. With the theorem, we observe that the palindrome must occur in the first $\mathbf{p}_f[1..Cn]$, so we only need the first $\lceil \log_2(Cn) \rceil$ instructions, or about $\log_2 C + O(1)$ past the end of n. Suppose we have a DFA M for the language associated with the predicate "the factor $\mathbf{p}_f[i..i+n-1]$ is a palindrome". We can quantify over $i \leq Cn$ by deleting the corresponding input and determinizing, and the resulting DFA T will be accurately decide whether there exists a palindrome of length n as long as we feed it a representation of n

with at least $\log_2 C + O(1)$ leading zeros. We can then fix T by a procedure similar to the one in Theorem 3.6, by considering the output of T when we add $\log_2 C + O(1)$ leading zeros to n and $\log_2 C + O(1)$ additional instructions to f.

In a personal communication, Shallit claims that all the example queries in the previous section can be solved with DFAs in this way. It is an open problem to characterize which queries we can mechanically decide with DFAs. It is crucial that the length of the quantified variable is bounded by the lengths of the remaining variables plus a constant, but we are unable to prove that this is sufficient.

Chapter 5

Abelian and Additive Powers

This section covers abelian powers and additive powers. These powers are like ordinary powers in that they are sequences of consecutive, equivalent words, but with a weaker notion of equivalence than equality. Abelian powers provide our first example of a query that we provably *cannot* express as a logical formula in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$. Nevertheless, our decidability results are frequently useful in abelian and additive power avoidance problems.

5.1 Definition and Notation

Definition 5.1. We say two words $x, y \in \Sigma^*$ are *abelian equivalent* if we can permute the symbols in one to obtain the other. Equivalently, x is abelian equivalent to y if $\psi(x) = \psi(y)$.

An *abelian dth power* is a finite word $x \in \Sigma^*$ of the form $x = x_1 \cdots x_d$ where the x_is are pairwise abelian equivalent.

Definition 5.2. We say two words $x, y \in \mathbb{N}^*$ are *additively equivalent* if $\sum x = \sum y$ and $|x| = |y|$.

An *additive dth power* is a finite word $x \in \mathbb{N}^*$ of the form $x = x_1 \cdots x_d$ where the x_is are pairwise additively equivalent.

5.2 Inexpressibility

Abelian powers are not as susceptible to the predicate techniques as ordinary powers. Intuitively, it is easy to test if two factors are identical, but difficult to check for abelian

equivalence. We will prove that the set of occurrences of abelian squares (or higher abelian powers) in the paperfolding word is not automatic. This proves that there is no general formula for the query "is $w[i..j]$ an abelian square?" in our logical framework. For certain automatic sequences, we show that the set of occurrences of abelian squares is not even expressible in $\mathsf{FO}(\mathbb{N}, <, +, V_k)$.

Recall the paperfolding word, (from Section 4.2)

$$\mathbf{p} = \mathbf{p}[1..\infty] = 110110011100100\cdots,$$

given by iterating the map $f(w) = w1\overline{w}^R$.

Definition 5.3. Given an infinite binary word $w = w[1..\infty] \in \{0,1\}^*$, define a function $\Delta_w \colon \mathbb{N} \to \mathbb{Z}$ such that $\Delta_w(i) = |w[1..i]|_1 - |w[1..i]|_0$.

Theorem 5.4. *Let $i \in \mathbb{N}$ be a natural number, and suppose $z \in \{0,1\}^*$ is a binary representation for i with at least one leading zero. Then*

$$\Delta_{\mathbf{p}}(i) = |z|_{01} + |z|_{10}\,.$$

Proof. Suppose that $i = 2^k + j$ for some $0 \le j < 2^{k-1}$. That is, 2^k is the place value of the leading one. Given the binary representation for i, we may obtain the binary representation for j by deleting the most significant one. Then the binary representation for $2^k - 1 - j$ is the digit-wise complement of j, since the binary representation of $j + (2^k - 1 - j) = 2^k - 1$ is all ones. If we let z' be a binary representation for $2^k - 1 - j$ with at least one leading zero, then it is not hard to see that $|z|_{01} + |z|_{10} = 1 + |z'|_{01} + |z'|_{10}$. If we can show that $\Delta_{\mathbf{p}}(i) = 1 + \Delta_{\mathbf{p}}(2^k - 1 - j)$ then a simple induction completes the proof.

Let us show that $\Delta_{\mathbf{p}}(i) = |z|_{01} + |z|_{10}$ by induction on i. If $i = 0$ then clearly $\Delta_{\mathbf{p}}(0) = 0 = |z|_{01} + |z|_{10}$.

If $i > 0$ then we can write $i = 2^k + j$ for some $k \ge 0$ and $0 \le j < 2^k$. Recall that the paperfolding word can be obtained by iterating $f(w) = w1\overline{w}^R$, starting at the empty word. In particular, $f^{(n)}(\varepsilon)$ is the length n prefix of $\mathbf{p}$, $w := \mathbf{p}[1..2^n - 1]$. Since $i = 2^k + j < 2^{k+1}$, it is contained in $f(w) = w1\overline{w}^R$. Then $\mathbf{p}[1..i]$ is of the form $uv1\overline{v}^R$, where $w = uv$. Notice that $|u| = 2^k - 1 - j$ and $|v| = j$. Hence,

$$\begin{aligned}
\Delta_{\mathbf{p}}(i) &= |\mathbf{p}[1..i]|_1 - |\mathbf{p}[1..i]|_0 \\
&= \left|uv1\overline{v}^R\right|_1 - \left|uv1\overline{v}^R\right|_0 \\
&= 1 + |u|_1 - |u|_0 \\
&= 1 + \left|\mathbf{p}[1..2^k - 1 - j]\right|_1 - \left|\mathbf{p}[1..2^k - 1 - j]\right|_0 \\
&= 1 + \Delta_{\mathbf{p}}(2^k - 1 - j).
\end{aligned}$$

By the induction hypothesis, $\Delta_{\mathbf{p}}(2^k - 1 - j) = |z'|_{01} + |z'|_{10}$, where z' is a binary representation for $2^k - 1 - j$ with at least one leading zero. Hence, it suffices to show that

$$|z|_{01} + |z|_{10} = 1 + |z'|_{01} + |z'|_{10}\,.$$

Assume without loss of generality that $|z| = |z'| \geq k + 1$. Note that if we add $i = 2^k + j$ and $2^k - 1 - j$, we obtain $2^{k+1} - 1$. The least significant $k + 1$ bits of $2^{k+1} - 1$ are all ones, and the only way this can happen is if the least significant $k + 1$ bits of z and z' are complementary. That is, $z[n] = \overline{z'[n]}$ for all the least significant $k + 1$ positions. In other words, we obtain z' by flipping the least significant $k + 1$ bits of z. This flips every occurrence of 01 to 10 and vice versa, except for the most significant occurrence of 01, which is removed. Hence

$$\begin{aligned} |z|_{01} &= |z'|_{10} + 1 \\ |z|_{10} &= |z'|_{01} \end{aligned}$$

which completes the proof. □

Theorem 5.5. *The set*

$$A = \{(i, j, k) : \mathbf{p}[i + 1..k] \text{ is an abelian square and } i + k = 2j\}$$

is not 2*-automatic.*

Proof. Suppose that A is 2-automatic. Then

$$L := \{(a)_2 : a \in A\} \subseteq (\{0, 1\}^3)^*$$

is a regular language. Define a morphism $h\colon \{\mathtt{a}, \mathtt{b}\}^* \to (\{\mathtt{0}, \mathtt{1}\}^3)^*$ such that

$$h(\mathtt{a}) = \begin{matrix} \mathtt{00000} \\ \mathtt{00100} \\ \mathtt{01000} \end{matrix} \qquad h(\mathtt{b}) = \begin{matrix} \mathtt{00010} \\ \mathtt{00110} \\ \mathtt{01010} \end{matrix},$$

where the columns represent triples in $\{\mathtt{0}, \mathtt{1}\}^3$. We will call $h(\mathtt{a})$ and $h(\mathtt{b})$ *blocks*, and think of a word $h(x)$ as composed of blocks.

Let $x \in \{\mathtt{a}, \mathtt{b}\}^*$ and suppose that $(i, j, k) = \langle h(x) \rangle_2 \in \mathbb{N}^3$. It is not hard to check that $i + k = 2j$, simply by performing the addition a block at a time (there are no carries from one block to another).

Each block starts and ends in 0 in each coordinate, so any occurrences of 01 or 10 are within blocks. Hence,

$$\begin{aligned}\Delta_{\mathbf{p}}(i) &= 2\,|x|_{\mathtt{b}} \\ \Delta_{\mathbf{p}}(j) &= 2\,|x|_{\mathtt{a}} + 2\,|x|_{\mathtt{b}} \\ \Delta_{\mathbf{p}}(k) &= 2\,|x|_{\mathtt{a}} + 4\,|x|_{\mathtt{b}}\,.\end{aligned}$$

Then $\mathbf{p}[i+1..k]$ is an abelian square if and only if $\Delta_{\mathbf{p}}(i), \Delta_{\mathbf{p}}(j), \Delta_{\mathbf{p}}(k)$ is an arithmetic progression. That is, when

$$\begin{aligned}\Delta_{\mathbf{p}}(i) + \Delta_{\mathbf{p}}(k) &= 2\Delta_{\mathbf{p}}(j) \\ 2\,|x|_{\mathtt{a}} + 6\,|x|_{\mathtt{b}} &= 4\,|x|_{\mathtt{a}} + 4\,|x|_{\mathtt{b}} \\ |x|_{\mathtt{a}} &= |x|_{\mathtt{b}}\end{aligned}$$

Therefore,

$$h^{-1}(L) = \{x \in \{\mathtt{a},\mathtt{b}\}^* : |x|_{\mathtt{a}} = |x|_{\mathtt{b}}\}$$

which is a textbook non-regular language. On the other hand, L is a regular language, and therefore $h^{-1}(L)$ is regular. □

Corollary 5.6. *The set*

$$A = \{(i,j,k) \in \mathbb{N}^3 : \mathbf{p}[i+1..k] \text{ is an abelian square and } i+k = 2j\}$$

is not k-automatic for any $k \geq 2$.

Proof. Suppose that A is k-automatic, for some $k \geq 2$. By the previous theorem, k is not multiplicatively dependent with 2, since that would make A a 2-automatic set.

If A is k-automatic then we can constructively show that

$$B := \{i \in \mathbb{N} : \mathbf{p}[i+1..i+2] \text{ is an abelian square}\}$$

is also k-automatic by adding the condition $k = i+2$, then quantifying out j and k. But $\mathbf{p}[i+1..i+2]$ is an abelian square if and only if $\mathbf{p}[i+1] = \mathbf{p}[i+2]$. Then the sequence $\mathbf{q} = \mathbf{q}[1..\infty] \in \{\mathtt{0},\mathtt{1}\}^\omega$ where

$$\mathbf{q}[i] = (\mathbf{p}[i+1] - \mathbf{p}[i]) \bmod 2$$

is k-automatic. That is, $\mathbf{q}$ is the sequence of first differences of $\mathbf{p}$, modulo 2. There is a DFAO that computes the running sum modulo 2, so we can apply Theorem 3.14 to recover $\mathbf{p}$ from $\mathbf{q}$, and hence $\mathbf{p}$ is k-automatic.

Of course, $\mathbf{p}$ is 2-automatic and aperiodic, so by Theorem 2.15, k and 2 are multiplicatively dependent. But we already know k and 2 are not multiplicatively dependent, so by contradiction, A is not k-automatic for any $k \geq 2$. □

As we noted earlier, there cannot be a general formula that describes abelian squares, and for some automatic sequences we cannot even give a formula specific to that sequence. However, for many k-automatic sequences that arise in practice, the set of occurrences of abelian squares is k-automatic. .

5.3 Counting Symbols with Automata

In this section, we suppose that we are given an automaton that tells us how many times each symbol occurs in a given prefix of our automatic sequence. For example, the Thue-Morse sequence is clearly composed of 01 and 10 blocks. Hence, the number of 0s and number of 1s in a given prefix are either equal, or differ by 1. The DFAO in Figure 5.1 shows that function $\Delta_{\mathbf{t}}(n) = |\mathbf{t}[0..n-1]|_1 - |\mathbf{t}[0..n-1]|_0$ is 2-automatic.

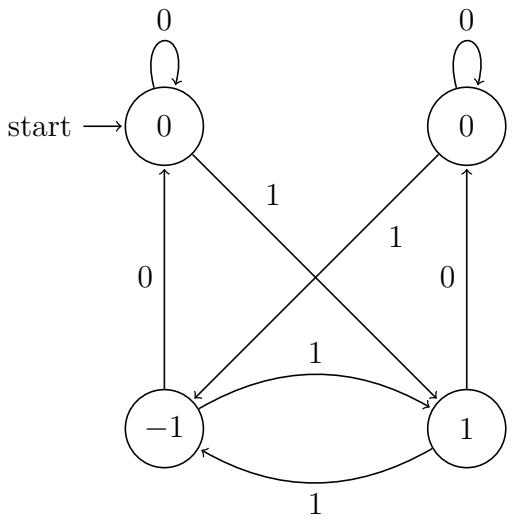


Figure 5.1: Automaton for computing $\Delta_{\mathbf{t}}(n)$, given n in binary.

If w is a k-automatic binary word such that $\Delta_w(n)$ is k-automatic, then it follows that

the set of occurrences of abelian squares in w,

$$\{(a,b,c) \in \mathbb{N}^3 : \Delta_w(a) + \Delta_w(c) = 2\Delta_w(b), a + c = 2b\},$$

is also k-automatic, and similarly for higher abelian powers.

The following theorem generalizes this idea to words with larger alphabets, and where the limiting frequencies of symbols are not necessarily uniform.

Theorem 5.7. *Let $w \in \Sigma^\omega$ be a k-automatic word. Suppose there exists a vector $\nu \in \mathbb{R}^{|\Sigma|}$ such that $\psi(w[0..n-1]) - n\nu$ is bounded. Then the sequence $(\psi(w[0..n-1]) - n\nu)_{n=0}^\infty$ is k-automatic.*

Proof. Note that

$$\lim_{n\to\infty} \frac{\|\psi(w[0..n-1]) - n\nu\|}{n} = 0$$

because the numerator is bounded. It follows that $\lim_{n\to\infty} \psi(w[0..n-1])/n = \nu$, so ν is the limiting frequency vector. Since w is a k-automatic sequence, the limiting frequency of each symbol (if it exists) is rational (see [6, Theorem 8.4.5, p. 268]). Then ν is a rational vector, which we can express as $\frac{\nu'}{d}$ for $\nu' \in\in \mathbb{N}^{|\Sigma|}$ an integer vector and some integer $d > 0$. Observe that $d\psi(w[0..n-1]) - n\nu'$ is an integer vector, and it is bounded, so it belongs to a finite set of vectors. This establishes that $(\psi(w[0..n-1]) - n\nu)_{n=0}^\infty$ is a sequence over a finite alphabet.

Now consider the first differences.

$$(\psi(w[0..n]) - (n+1)\nu) - (\psi(w[0..n-1]) - n\nu) = \psi(w[n]) - \nu$$

It follows that the first difference sequence is a coding of w, and therefore k-automatic. Then $(\psi(w[0..n-1]) - n\nu)_{n=0}^\infty$ is k-regular by Corollary 2.20. Since $(\psi(w[0..n-1]) - n\nu)_{n=0}^\infty$ is k-regular and bounded, Theorem 2.18 implies that it is k-automatic. □

This result is of limited use if we hope to prove a k-automatic sequence avoids abelian d-th powers, since the following the result by Au, Robertson and Shallit shows that such a word cannot avoid abelian powers.

Theorem 5.8. *Let $w \in \Sigma^\omega$ be a k-automatic word. Suppose there exists a vector $\nu \in \mathbb{Q}^{|\Sigma|}$ such that $\psi(w[0..n-1]) - n\nu$ is bounded. Then w contains abelian dth-powers for all $d \geq 0$.*

Proof. See [7, Theorem 2]. □

Synchronized Functions

What happens when $\psi(w[0..n-1]) - n\nu$ is not bounded? For example, consider the binary word $\mathbf{z} = 010^2 1^4 0^8 1^{16} \dots$ where $\mathbf{z}[i]$ is the parity of the length of the canonical representation of i (with the convention that 0 is represented by ε). It is not difficult to show that $\Delta_{\mathbf{z}}(n)$ achieves $-\frac{n}{3} + O(1)$ and $\frac{n}{3} + O(1)$ infinitely many times. Clearly $\Delta_{\mathbf{z}}(n)$ is not k-automatic.

Recall from Section 2.4.2 that a function $f\colon \mathbb{N} \to \mathbb{N}$ is k-synchronized if the graph, $\{(n, f(n)) : n \in \mathbb{N}\}$, is k-automatic. Since the graph is k-automatic, we can add a predicate F to $\mathsf{FO}(\mathbb{N}, <, +, V_k, \{P_a\}_{a\in\Gamma})$ such that $F(n, m)$ is true if and only if $m = f(n)$. This allows us to express queries involving that function. In particular, if $n \mapsto \psi(w[0..n-1])$ is a k-synchronized then we can use it to express queries about abelian and additive powers.

For instance, consider the graph of $\Delta_z(n)$,

$$S = \{(n, \Delta_z(n)) : n \in \mathbb{N}\}.$$

With some attention to detail, one can show that

$$\Delta_z(n) = \begin{cases} 0, & \text{if } n = 0; \\ \frac{-4^k-2}{3}, & \text{if } n = 4^k; \\ \frac{2\cdot 4^k-2}{3}, & \text{if } n = 2\cdot 4^k; \\ \Delta_z(4^k) + n - 4^k, & \text{if } 4^k < n < 2\cdot 4^k; \\ \Delta_z(4^k) - n + 2\cdot 4^k, & \text{if } 2\cdot 4^k < n < 4\cdot 4^k. \end{cases}$$

We have seen how to find the largest power of 4 less than n in the first-order theory $\mathsf{FO}(\mathbb{N}, <, +, V_2)$. We can also add/subtract/multiply/divide by constants, so we can compute $\frac{2\cdot 4^k-2}{3}$. Hence, there is a predicate for the set S in $\mathsf{FO}(\mathbb{N}, <, +, V_2)$. Therefore S is 2-automatic, and $\Delta_z(n)$ is 2-synchronized. It follows that the map $n \mapsto \psi(\mathbf{z}[0..n-1])$ is 2-synchronized because

$$|\mathbf{z}[0..n-1]|_0 = \frac{1}{2}\left(n + \Delta_z(n)\right)$$
$$|\mathbf{z}[0..n-1]|_1 = \frac{1}{2}\left(n - \Delta_z(n)\right).$$

Then $\psi(\mathbf{z}[i..j-1]) = \psi(\mathbf{z}[j..k-1])$ if and only if

$$\exists x_i, x_j, x_k, y_i, y_j, y_k \; P(i, x_i, y_i) \wedge P(i, x_j, y_j) \wedge P(k, x_k, y_k) \wedge (2x_j = x_i + x_k) \wedge (2y_j = y_i + y_k)$$

Therefore the set of occurrences of abelian squares in $\mathbf{z}$ is 2-automatic, and similarly for higher abelian powers.

5.3.1 Linear Algebra and Abelian Properties

Recall that for a pure morphic word $w = \varphi^\omega(c) \in \Gamma^\omega$, we have a morphic numeration system where the representation of n is connected to the morphic decomposition,

$$w[0..n-1] = \varphi(\varphi(\cdots\varphi(\varphi(c_k)c_{k-1})\cdots c_2)c_1)c_0$$

for short words $c_0, \ldots, c_k \in \Delta_\varphi \subseteq \Gamma^*$.

Now observe that $\psi(xy) = \psi(x) + \psi(y)$ because ψ is a homomorphism, and $\psi(\varphi(x)) = M\psi(x)$ for some matrix M. We see that the columns of M must be $\psi(\varphi(c))$ for $c \in \Gamma$, so $M \in \mathbb{N}^{n\times n}$ (where $n = |\Gamma|$) is the *incidence matrix* of φ. Using these two properties, we see that

$$\begin{aligned}\psi(w[0..n-1]) &= M\left(M\left(\cdots M\left(M\psi(c_k) + \psi(c_{k-1})\right) + \psi(c_2)\right) + \psi(c_1)\right) + \psi(c_0)\\ &= \sum_{i=0}^{k} M^i\psi(c_i)\end{aligned}$$

Repeated multiplication by M leads us to consider an eigendecomposition of M. Specifically, let $M = P\Lambda P^{-1}$ where $\Lambda \in \mathbb{C}^{n\times n}$ is the Jordan normal form of M, and $P \in \mathbb{C}^{n\times n}$ is invertible. Then let $\tau\colon \Gamma^* \to \mathbb{C}^n$ be the map $x \mapsto P^{-1}\psi(x)$, and note that

$$\begin{aligned}\tau(w[0..n-1]) &= P^{-1}\psi(w[0..n-1])\\ &= P^{-1}\sum_{i=0}^{k}(P\Lambda P^{-1})^i\psi(c_i)\\ &= \sum_{i=0}^{k}\Lambda^i P^{-1}\psi(c_i)\\ &= \sum_{i=0}^{k}\Lambda^i\tau(c_i).\end{aligned}$$

Then the structure of Λ tells us about the Parikh vectors corresponding to factors of w, which helps us describe the set of abelian powers. In some cases, we can use it to show that $n \mapsto \tau(w[0..n-1])$ is semi-synchronized, and note that $\tau(x) = \tau(y)$ if and only if $\psi(x) = \psi(y)$ since τ is invertible. In other cases, we give a direct method (i.e., without building a semi-synchronized automaton) for proving morphic words avoid abelian squares.

5.3.2 First Example

For example, let $\Gamma = \{0, 1, 2, 3\}$ and consider the word $\mathbf{q} \in \Gamma^\omega$ defined as the fixed point of $\varphi\colon \Gamma^* \to \Gamma^*$ where

$$\begin{aligned}\varphi(0) &= 01\\ \varphi(1) &= 12\\ \varphi(2) &= 23\\ \varphi(3) &= 30.\end{aligned}$$

The incidence matrix of φ is

$$M = \begin{pmatrix}1 & 0 & 0 & 1\\ 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 1\end{pmatrix}.$$

However, the incidence matrix of φ^4 is M^4, and it turns out that M^4 has integer eigenvalues 16, -4 (with multiplicity 2) and 0. Since $\mathbf{q}$ is a fixed point of φ^4, we will work with φ^4 instead of φ. Decompose M^4 as $P\Lambda P^{-1}$ where $P, \Lambda \in \mathbb{Q}^{4\times 4}$ are as follows

$$P = \frac{1}{4}\begin{pmatrix}1 & 1 & 0 & 1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & 1\\ 1 & 0 & 1 & -1\end{pmatrix} \qquad \Lambda = \begin{pmatrix}16 & 0 & 0 & 0\\ 0 & -4 & 0 & 0\\ 0 & 0 & -4 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}$$

Define $\tau\colon \Gamma^* \to \mathbb{N}^4$ such that $\tau(x) = P^{-1}\psi(x)$. We claim that $n \mapsto \tau(\mathbf{q}[0..n-1])$ is semi-synchronized, over base 16 and base -4. We will prove this, but first we need a technical lemma about conversion between representations in base r.

Lemma 5.9. *Let $r \leq 2$ be an integer. and let $S \subseteq \mathbb{Z}$ be a finite set of integers. We define $\langle\cdot\rangle_{-r} : (\Sigma_r \cup S)^* \to \mathbb{N}$ where*

$$\langle a[0]a[1]\cdots a[n-1]\rangle_{-r} := \sum_{i=0}^{n-1} a[i](-r)^i.$$

Then the language

$$L = \{(u, v) \in (S \times \Sigma_r)^* : \langle u\rangle_{-r} = \langle v\rangle_{-r}\}$$

is regular.

Proof. We construct a DFA for the language L where the set of states, Q, is a finite subset of integers and a dead state q^*. For now, assume that Q contains *all* integers; we will bound Q to a finite set. Suppose that $(u, v) \in (S \times \Sigma_r)^*$ is a word in L. Consider the equation $\langle u \rangle_{-r} = \langle v \rangle_{-r}$ modulo r^k for all $0 \leq k \leq n$.

$$\sum_{i=0}^{k-1} u[i](-r)^i \equiv \sum_{i=0}^{k-1} v[i](-r)^i \pmod{r^k} \tag{5.1}$$

We design our automaton to assume (5.1) holds for k, and using information in the current state and the values of $u[k]$ and $v[k]$, decide whether (5.1) holds for $k+1$. If not, we transition to the dead state. Otherwise, we transition to a state such that we preserve the invariant in (5.2).

$$\delta(q_0, (u[0..k-1], v[0..k-1]))(-r)^k = \sum_{i=0}^{k-1} (u[i] - v[i])(-r)^i \tag{5.2}$$

Observe that when $k = 0$, (5.2) implies that $q_0 = 0$. If we let $\delta(q_0, (u[0..k-1], v[0..k-1]))$ be state $q \in Q$, then

$$\begin{aligned}
\delta(q, (u[k], v[k]))(-r)^{k+1} &= \delta(q_0, (u[0..k], v[0..k]))(-r)^{k+1} \\
&= \sum_{i=0}^{k} (u[i] - v[i])(-r)^i \\
&= \sum_{i=0}^{k-1} (u[i] - v[i])(-r)^i + (u[k] - v[k])(-r)^k \\
&= q(-r)^k + (u[k] - v[k])(-r)^k,
\end{aligned}$$

so $\delta(q, (u[k] - v[k]))(-r) = q + u[k] - v[k]$. Hence, we define our automaton so that there is a transition from q to q' on input (a, b) if and only if $-rq' = q + a - b$, and this will preserve the invariant (5.2). Let all other inputs transition to the dead state, q^*. It is clear from (5.2) that $\langle u \rangle_{-r} = \langle v \rangle_{-r}$ if and only if we end in state 0, so let 0 be the sole final state.

All that is left is to show that we can only reach a finite set of states starting from 0. Let $B = \max_{s \in S} |s|$ be an upper bound for values in S. We will show by induction that any reachable state in $Q \backslash \{q^*\}$ is at most $\frac{B}{r-1} + 1$. Note that bounds holds for $q_0 = 0$, and

any other reachable state q' is of the form $\delta(q,(a,b))$ for some reachable state q. Then

$$\begin{aligned} q'(-r) &= q + a - b \\ |q'|r &\leq |q| + |a| + |b| \\ &\leq \frac{B}{r-1} + 1 + B + r - 1 \\ &= r\left(\frac{B}{r-1} + 1\right) \\ |q'| &\leq \frac{B}{r-1} + 1. \end{aligned}$$

This completes the induction, and the proof. □

Theorem 5.10. *Let φ, $\mathbf{q}$, P, Λ and τ be defined as above. Then $n \mapsto \tau(\mathbf{q}[0..n-1])$ is semi-synchronized where the input and first output coordinate are in base 16, the next two coordinates are in base -4, and any numeration system for the last coordinate.*

Proof. The natural numeration system for a fixed point of φ^4 is base-16, since φ^4 is 16-uniform. Theorem 2.11 says that $L \subseteq \Delta_{\varphi^4}$, the language of all morphic decompositions of $\mathbf{q}$, is regular. Consider the coding $h\colon \Delta_{\varphi^4}^* \to (\Sigma_{16} \times \Delta_{\varphi^4})^*$ such that $h(a) = (a, |a|)$. Then $h(L)$ is also regular. It follows that the function from a base-16 representation $d = d[0..k-1] \in \Sigma_{16}$ (for an integer n) to the unique morphic decomposition $c = c[0..k-1] \in \Delta_{\varphi^4}^*$ for the prefix $\mathbf{q}[0..n-1]$, is a synchronized function, since $h(L)$ accepts (d, c) (and in general, the graph of the function).

Therefore, we will assume that we are given the morphic decomposition $c = c[0..k-1]$ for $\mathbf{q}[0..n-1]$, since we can combine synchronized functions with predicates. Now recall the equation

$$\tau(\mathbf{q}[0..n-1]) = \sum_{i=0}^{k-1} \Lambda^i \tau(c[i])$$

for a morphic decomposition. For each coordinate $j = 1, 2, 3, 4$, we have

$$\tau(\mathbf{q}[0..n-1])^{(j)} = \sum_{i=0}^{k-1} \lambda_j^i \tau(c[i])^{(j)}.$$

where $\lambda_1 = 16$, $\lambda_2 = \lambda_3 = -4$ and $\lambda_4 = 0$ are the eigenvalues (along the diagonal of Λ). It suffices to show that some automaton computes $\tau(\mathbf{q}[0..n-1])^{(j)}$ from $c[0], \ldots, c[k-1] \in \Delta_{\varphi^4}$.

In the first coordinate, it turns out [1] that $\tau(c[i])^{(1)}$ is the sum of the entries in $\psi(c[i])$. The sum of the entries in $\psi(c[i])$ is just $|c[i]|$, so

$$\tau(\mathbf{q}[0..n-1]) = \sum_{i=0}^{n-1} 16^i \, |c[i]| = n.$$

It is therefore trivial to compute the first coordinate given a base-16 representation for n.

In the fourth coordinate, all terms vanish except for $\tau(c[0])^{(4)}$ because $\lambda_4 = 0$. Therefore we can compute $\tau(\mathbf{q}[0..n-1])^{(4)}$ in any numeration system, given n in base 16.

Finally, observe that the second and third coordinates are of the form

$$\sum_{i=0}^{n-1} (-4)^i \tau(c[i])^{(j)} = \langle c \rangle_{-4}$$

for $j = 2, 3$. Then $\tau(c[i])^{(j)}$ is in the finite set

$$S_j = \{\tau(c)^{(j)} : c \in \Delta_{\varphi^4}\}$$

This is precisely the situation where Lemma 5.9 applies, with $r = 4$ and $S = S_j$. This gives us (with some minor remapping of the input) an automaton that converts from c to the canonical base-(-4) representation of $\langle c \rangle_{-4}$. Hence, the second and third coordinates are synchronized. □

Using the fact that $n \mapsto \tau(\mathbf{q}[0..n-1])$ is semi-synchronized, we can given a fairly mechanical proof of the following theorem.

Theorem 5.11. *There are no abelian cubes in* $\mathbf{q}$.

Proof. Observe that there is an abelian cube in $\mathbf{q}$ if we can find i, j, k, ℓ such that

$$\psi(\mathbf{q}[0..j-1]) - \psi(\mathbf{q}[0..i-1]) = \psi(\mathbf{q}[0..k-1]) - \psi(\mathbf{q}[0..j-1]) = \psi(\mathbf{q}[0..\ell-1]) - \psi(\mathbf{q}[0..k-1]) \,.$$

Of course, $\psi(x) = \psi(y)$ if and only if $\tau(x) = \tau(y)$. Since $n \mapsto \tau(\mathbf{q}[0..n-1])$ is semi-synchronized, and addition is automatic in base-16 and base-(-4), we can construct an automaton that decides if

$$\tau(\mathbf{q}[0..j-1]) - \tau(\mathbf{q}[0..i-1]) = \tau(\mathbf{q}[0..k-1]) - \tau(\mathbf{q}[0..j-1]) = \tau(\mathbf{q}[0..\ell-1]) - \tau(\mathbf{q}[0..k-1]).$$

The automaton rejects all inputs, so we conclude that there are no abelian cubes in $\mathbf{q}$. □

[1] This will always happen when φ is k-uniform, and is related to the fact that $[1, \ldots, 1]$ is a left eigenvector for the incidence matrix with eigenvalue k.

5.3.3 Second Example

For our second example is taken from [15]. Let $\Sigma = \{0, 1, 3, 4\} \subseteq \mathbb{N}$ and define a morphism $\varphi\colon \Sigma^* \to \Sigma^*$ where

$$\begin{aligned}\varphi(0) &= 03\\ \varphi(1) &= 43\\ \varphi(3) &= 1\\ \varphi(4) &= 01.\end{aligned}$$

Let $w = \varphi^\omega(0) \in \Sigma^\omega$ be the fixed point of φ.

Cassaigne et al. prove the following theorem about w in [15].

Theorem 5.12. *There are no additive cubes in w.*

We give a rough sketch of their approach, and discuss how similar ideas might be applied to abelian/additive problems in other morphic words.

Recall that there is a morphic numeration system for φ over Σ_2. Since we will be working directly with the morphic decompositions, we prefer the numeration system $\mathcal{N} = (\Gamma, L, \langle\cdot\rangle_{\mathcal{N}})$ where $\Gamma = \Delta_\varphi = \{\varepsilon, 0, 4\}$ (recall that Δ_φ is a set of words), L is the language of morphic decompositions (most significant "digit" first), and $\langle\cdot\rangle_{\mathcal{N}}$ maps a morphic decomposition to the length of the prefix it decomposes. Note that L is a regular language by Theorem 2.11, and L is accepted by the DFA in Figure 5.2. Note that $\mathcal{N}$ is essentially the morphic numeration system; if we apply the coding $\varepsilon \mapsto 0, 0 \mapsto 1, 4 \mapsto 1$ to L, we obtain the set of representations for the morphic numeration system.

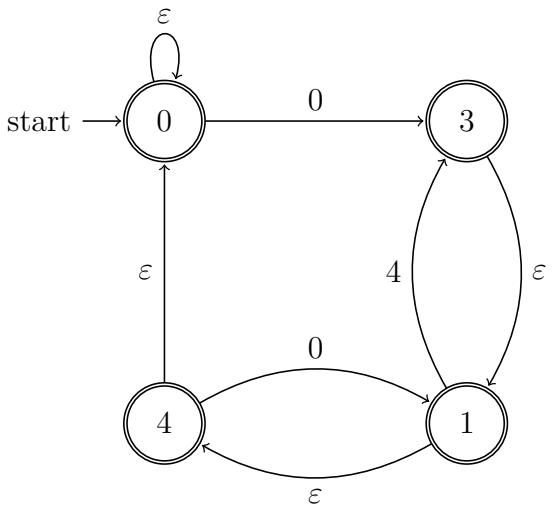


Figure 5.2: DFA for L.

There is an additive cube in w if and only if there exist $i_0, i_1, i_2, i_3 \in \mathbb{N}$ such that

- $i_0 < i_1 < i_2 < i_3$,
- $|w[i_0..i_1 - 1]| = |w[i_1..i_2 - 1]| = |w[i_2..i_3 - 1]|$, and
- $\sum w[i_0..i_1 - 1] = \sum w[i_1..i_2 - 1] = \sum w[i_2..i_3 - 1]$.

Note that the set

$$R := \{(i_0, i_1, i_2, i_3) \in \mathbb{N}^4 : i_0 < i_1 < i_2 < i_3\}$$

is $\mathcal{N}$-automatic, which covers the first condition. The length and sum conditions we must check separately. To do this we consider Parikh vectors, and the first step is to relate $\psi(w[0..n-1])$ to $(n)_{\mathcal{N}}$, the $\mathcal{N}$ representation for n.

Let $M \in \mathbb{N}^{4\times 4}$ be the incidence matrix of φ

$$M = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

We decompose M as $P\Lambda P^{-1}$ again, for $P, \Lambda \in \mathbb{C}^{4\times 4}$. We let $\tau\colon \Sigma^* \to \mathbb{C}^4$ be such that $\tau(x) = P^{-1}\psi(x)$, as before. Unfortunately, the eigenvalues of M are all rational, or even

real.

$$\begin{aligned}\lambda_1 &\doteq 1.69028\\ \lambda_2 &\doteq -1.50507\\ \lambda_3 &\doteq 0.40739 + 0.47657i\\ \lambda_4 &\doteq 0.40739 - 0.47657i.\end{aligned}$$

As far as we know, there is no choice of numeration systems such that $n \mapsto \tau(w[0..n-1])$ is a semi-synchronized function, so we cannot proceed as we did in the first example. Instead, we have the following theorem.

Theorem 5.13. *Let $c = c[0..m-1] \in L$ be an MSD-first $\mathcal{N}$ representation for an integer $n \geq 0$. Then there is a sequence of vectors $v_0, v_1, \ldots, v_m \in \mathbb{C}^4$ such that $v_0 = (0,0,0,0)$, $v_m = \tau(w[0..n-1])$ and $v_{i+1} = \Lambda v_i + \tau(c[i])$.*

Proof. Each prefix of c corresponds to a representation, since it is MSD-first.

$$\begin{aligned}n_0 &= 0\\ n_1 &= \langle c[0]\rangle_{\mathcal{N}}\\ n_2 &= \langle c[0..1]\rangle_{\mathcal{N}}\\ &\vdots\\ n_m &= \langle c[0..m-1]\rangle_{\mathcal{N}}\end{aligned}$$

and $n = n_m$ is the number represented by c. Since $\mathcal{N}$ is based on morphic decompositions, we have the following equation for length n_i and n_{i+1} prefixes.

$$w[0..n_{i+1}-1] = \varphi(w[0..n_i-1])c[i].$$

Now take $v_i = \tau(w[0..n_i-1])$ and note that

$$\begin{aligned}v_{i+1} &= \tau(w[0..n_{i+1}-1])\\ &= P^{-1}\psi(w[0..n_{i+1}-1])\\ &= P^{-1}\psi(\varphi(w[0..n_i-1])c[i])\\ &= P^{-1}M\psi(w[0..n_i-1]) + P^{-1}\psi(c[i])\\ &= \Lambda P^{-1}\psi(w[0..n_i-1]) + \tau(c[i])\\ &= \Lambda v_i + \tau(c[i]),\end{aligned}$$

as desired. □

Let $v_i^{(j)}$ denote the jth coordinate of v_i for $j = 1, 2, 3, 4$. The condition $v_{i+1} = \Lambda v_i + \tau(c[i])$ is equivalent to

$$v_{i+1}^{(j)} = \lambda_j v_i^{(j)} + \tau(c[i])^{(j)}$$

for all $j = 1, 2, 3, 4$. Since $c[i]$ comes from the finite set Γ, $\tau(c[i])^{(j)}$ is one of finitely many values, and is bounded above by some B_j. Let us abstract away from the problem at hand, and consider sequences with similar properties.

Definition 5.14. Let $(x_i)_{i=0}^m$ be a sequence of complex numbers such that $x_0 = 0$. We say $(x_i)_{i=0}^m$ is a *pseudo-geometric sequence* if there exists $\lambda \in \mathbb{C}$ and $B \in \mathbb{R}$ such that $|x_{i+1} - \lambda x_i| \leq B$ for all $0 \leq i < m$.

It is not surprising that pseudo-geometric sequences behave like geometric sequences in the following sense. If $|\lambda| < 1$ then exponential decay prevents the sequence from growing large. On the other hand, if $|\lambda| > 1$ then exponential growth will cause the sequence to explode unless it is kept below some threshold. We make these ideas precise in the following two propositions.

Proposition 5.15. *Let $(x_i)_{i=0}^m$ be a pseudo-geometric sequence, witnessed by $\lambda \in \mathbb{C}$ and $B \in \mathbb{R}$, and suppose $|\lambda| < 1$. Then $|x_i| < \frac{B}{1-|\lambda|}$ for all $i \geq 0$.*

Proof. We show that $|x_i| < \frac{B}{1-|\lambda|}$ by induction. Clearly the induction hypothesis holds for $i = 0$. For $i > 0$, we have

$$\begin{aligned} |x_{i+1}| &\leq |\lambda x_i| + |x_{i+1} - \lambda x_i| \\ &\leq |\lambda||x_i| + B \\ &< \frac{B|\lambda|}{1-|\lambda|} + B \\ &= \frac{B}{1-|\lambda|}. \end{aligned}$$

□

Proposition 5.16. *Let $(x_i)_{i=0}^m$ be a pseudo-geometric sequence, witnessed by $\lambda \in \mathbb{C}$ and $B \in \mathbb{R}$, and suppose $|\lambda| > 1$. If $|x_i| > C$ for some real $C \geq \frac{B}{|\lambda|-1}$ then $|x_j| > C$ for all $j \geq i$.*

Proof. We use induction to show that $|x_j| > C$ for $j \geq i$. The base case $j = i$ holds by assumption. Then for $j > i$ we have

$$\begin{aligned} |x_{i+1}| &\geq |\lambda x_i| - |x_{i+1} - \lambda x_i| \\ &\geq |\lambda||x_i| - B \\ &\geq |\lambda||x_i| - C(|\lambda| - 1) \\ &> |\lambda|C - C(|\lambda| - 1) \\ &= C. \end{aligned}$$

□

Using these two propositions, we give an upgraded version of Theorem 5.13.

Theorem 5.17. *Let $w[a_1 + 1..b_1]$ and $w[a_2 + 1..b_2]$ be a pair of factors in w such that*

$$\begin{aligned} |w[a_1 + 1..b_1]| &= |w[a_2 + 1..b_2]| \\ \sum w[a_1 + 1..b_1] &= \sum w[a_2 + 1..b_2]. \end{aligned}$$

Suppose $c \in (\Gamma^4)^*$ be a $\mathcal{N}$ representation for (a_1, a_2, b_1, b_2), and $|c| = m$. Then there exists a function $f \colon \Gamma^4 \to \mathbb{C}^4$ and a sequence of vectors $(v_j)_{j=0}^m$ such that $v_0 = 0$, $v_m = \tau(w[a_2..b_2 - 1]) - \tau(w[a_1..b_1 - 1])$, and $v_{j+1} = \Lambda v_j + f(c[j])$ is bounded. Furthermore, the vector lengths, $\|v_j\|$, are bounded by some universal (i.e., not dependent on a_1, a_2, b_1, b_2) constant B for all j.

Proof Sketch. We apply Theorem 5.13 four times, for each of the factors $w[0..a_1 - 1]$, $w[0..a_2 - 1]$, $w[0..b_1 - 1]$ and $w[0..b_2 - 1]$. The theorem gives us four sequences of vectors, and immediately consolidate them into a single sequence $(v_j)_{j=0}^m$ by taking a linear combination such that

$$v_m = \tau(w[0..a_1 - 1]) - \tau(w[0..a_2 - 1]) - \tau(w[0..b_1 - 1]) + \tau(w[0..b_2 - 1]).$$

Then $v_m = \tau(w[a_2..b_2 - 1]) - \tau(w[a_1..b_1 - 1])$, clearly $v_0 = 0$ and $v_{j+1} = \Lambda v_j + f(c[j])$, where $f(d_1, d_2, d_3, d_4) = \tau(d_1) - \tau(d_2) - \tau(d_3) + \tau(d_4)$.

Now observe that $(v_m^{(\ell)})_{j=0}^m$ is a pseudo-geometric sequence for each coordinate $\ell = 1, 2, 3, 4$, since $v_{j+1}^{(\ell)} - \lambda_\ell v_j^{(\ell)}$ is bounded. When $\ell = 3, 4$, Proposition 5.15 applies because $|\lambda_\ell| < 1$, so the third and fourth coordinates of v_j are bounded for all j.

The fact that two coordinates of v_j are bounded is like having two linear equations that v_j must satisfy. The constraints

$$|w[a_1+1..b_1]| = |w[a_2+1..b_2]|$$
$$\sum w[a_1+1..b_1] = \sum w[a_2+1..b_2]$$

give us two additional linear equations that v_m must satisfy. These 4 linear constraints on the 4-dimensional vector v_m are enough to show that $\|v_m\|$ is bounded by some constant C_0.

Now consider the first two coordinates of v_j. Proposition 5.16 applies because $|\lambda_\ell| < 1$ for $\ell = 1, 2$. If we choose the constant $C > C_0$ in Proposition 5.16, then since $|v_m^{(\ell)}| \leq C$, we have that $|v_j^{(\ell)}| \leq C$ for all j. For all j, we have bounded $|v_j^{(\ell)}|$ for each coordinate $\ell = 1, 2, 3, 4$, so $\|v_j\|$ is bounded. This completes the proof. □

We can now sketch a proof of our main result: w avoids additive cubes.

Proof Sketch. Suppose that w contains an additive cube, with blocks $w[i_0..i_1-1]$, $w[i_1..i_2-1]$ and $w[i_2..i_3-1]$ for $i_0, i_1, i_2, i_3 \in \mathbb{N}$. Apply Theorem 5.17 to $w[i_0..i_1-1]$ and $w[i_1..i_2-1]$, and then again to $w[i_1..i_2-1]$ and $w[i_2..i_3-1]$. If $\mathbf{c} \in (\Gamma^4)^*$ is the $\mathcal{N}$ representation for (i_0, i_1, i_2, i_3) and $|c| = m$, then this gives us two sequences of vectors, $(u_j)_{j=0}^m$ and $(v_j)_{j=0}^m$, and a pair of functions $f_1, f_2 \colon \Gamma^4 \to \mathbb{C}^4$ such that

- $u_0 = v_0 = 0$,
- $u_m = \tau(w[i_1..i_2-1]) - \tau(w[i_0..i_1-1])$ and $v_m = \tau(w[i_2..i_3-1]) - \tau(w[i_1..i_2-1])$,
- $u_{j+1} = \Lambda u_j + f_1(c[j])$ and $v_{j+1} = \Lambda v_j f_2(c[j])$, and
- u_j and v_j are bounded length vectors for all j.

Note that u_j is of the form $P^{-1}x_j$, for $x_j \in \mathbb{Z}^4$ an integer vector. Since u_j has bounded length, x_j also has bounded length, because P is invertible. There are only finitely many bounded length integer vectors, so u_j belongs to some finite set of possible vectors $X \subseteq \mathbb{C}^4$. Similarly, v_j belongs to X.

Hence, we can construct a finite automaton T that reads $c \in (\Gamma^4)^*$ representing $i_0 < i_1 < i_2 < i_3$, and keeps track of u_j and v_j, as long as they are within X. At the end of the input, u_m and v_m are encoded in the state of the automaton, so we can deduce whether

$$|w[i_0..i_1-1]| = |w[i_1..i_2-1]| = |w[i_2..i_3-1]|$$
$$\sum w[i_0..i_1-1] = \sum w[i_1..i_2-1] = \sum w[i_2..i_3-1]$$

from the state, and hence whether (i_0, i_1, i_2, i_3) delimits an additive cube. When we do this for w, we find that the automaton does not accept any words, so there are no additive cubes in w. □

Note that the proof in [15] proves better bounds than Proposition 5.15 and Proposition 5.16 in an effort to minimize the number of states in the finite automaton T. Even so, there are hundreds of thousands of states, so a computer is necessary to construct the automaton and test if it is empty.

5.3.4 Decidability for Abelian Powers

Consider the problem of finding abelian d-powers in a pure morphic word $w = \varphi^\omega(a)$. In many cases, the approach from the previous section will work.

- Compute the incidence matrix M of φ, and its eigendecomposition, $P\Lambda P^{-1}$.
- Show that a representation $c \in (\Gamma^{d+1})^*$ for d blocks leads to a sequence of vectors $(v_j)_{j=0}^m$ such that
  - $v_0 = 0$,
  - $v_{j+1} = \Lambda v_j + f(c)$ for some f from Γ^{d+1} to vectors, and
  - $v_m = 0$ if and only if c represents an abelian d-power.
- Argue that v_j has bounded length, because
  - if $|\lambda_\ell| < 1$ then coordinate ℓ is bounded by an analogue of Proposition 5.15,
  - if $|\lambda_\ell| > 1$ then coordinate ℓ is bounded by an analogue of Proposition 5.16.
- Construct a finite automaton that keeps track of v_j in the state as it reads the input $c \in (\Gamma^{d+1})^*$, and uses it to decide whether c represents an abelian d-power.

We can follow these steps to construct an automaton that accepts the set of occurrences of all abelian d-powers in a given morphic word $w = \varphi^\omega(a)$, with one important caveat. The method fails when some eigenvalue λ occurs with multiplicity or has unit modulus (i.e., $|\lambda| = 1$). Removing these caveats to obtain a decision procedure for abelian power avoidance is the subject of Problem 6.3. Note that Theorem 5.5 does not rule out the possibility of a decision procedure; it simply shows that we cannot always construct an automaton for the occurrences of abelian powers.

Compare this decision procedure to Currie and Rampersad's procedure in [22].

Theorem 5.18. *Let $w = \varphi^\omega(a) \in \Gamma^\omega$ be a morphic word. It is decidable whether w is abelian k-power free as long as*

- $|\varphi(\alpha)| > 1$ *for all* $\alpha \in \Gamma$,
- *the incidence matrix* M *of* φ *is nonsingular,*
- $\|M^{-1}\| < 1$ *where* $\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}$.

Their result is very similar to ours. In lieu of conditions on the eigenvalues of M, Currie and Rampersad require M to be nonsingular[2], and $\|M^{-1}\| < 1$. However, these conditions can be related to the eigenvalues as follows. A matrix is singular if and only if 0 is an eigenvalue, so the first condition says that 0 is not an eigenvalue of M. If x is an eigenvector of M and $\lambda \neq 0$ is the corresponding eigenvalue then

$$\left\|M^{-1}\right\| \geq \frac{\|M^{-1}x\|}{\|x\|} = \frac{|\lambda|^{-1}\|x\|}{\|x\|} = |\lambda|^{-1}.$$

The second condition says that $\|M^{-1}\| < 1$, so $|\lambda| > 1$ for all eigenvalues λ of M. Hence, their condition requires all eigenvalues to be outside the unit circle.

[2]In a personal communication with Currie, he suggested that this condition is not essential.

Chapter 6

Open Problems

We list a handful of open problems on decidability in automatic sequences.

6.1 Complexity Problems

First, a general question about the complexity of our algorithms.

Problem 6.1. What is the complexity of deciding sentences in $\mathsf{FO}(\mathbb{N}, <, \{P_a\}_{a\in\Gamma})$ about an automatic sequence $w \in \Gamma^\omega$?

Our bounds do not accurately reflect the practical complexity of our decidability algorithms. In some places, we can improve our crude upper bounds. For instance, the predicate "$i + j + k = n$" translates into "$(\exists m\,(i + j = m) \wedge (m + k = n))$", but the corresponding machine is much smaller than the bounds predicate. In general, the state complexity of a machine for "$i_1 + i_2 + \cdots + i_m = n$" grows logarithmically with m, the number of summands. Can we improve our bounds for other common subformulas? Testing two subwords for equality is probably the most common operation, so that seems like a good place to start.

Problem 6.2. Give nontrivial bounds for the state complexity of an automaton deciding "$w[i..i + n - 1] = w[j..j + n - 1]$", where $w \in \Gamma^\omega$ a k-automatic sequence.

6.2 Abelian Power Decidability

In Chapter 5, we outline an algorithm for deciding abelian k-power freeness of a given pure morphic word $w = \varphi^\omega(a) \in \Gamma^\omega$. Unfortunately, the algorithm only works if the eigenvalues of the incidence matrix of φ occur without multiplicity, and do not lie on the unit circle.

Problem 6.3. Let $w = \varphi^\omega(a) \in \Gamma^\omega$ be a pure morphic word. Suppose M is the incidence matrix of w. Can we extend our algorithm to decide whether w is abelian k-power free when

- M has repeated eigenvalues, or
- M has eigenvalues on the unit circle?

If not, can we extend the algorithm in the cases where

- all repeated eigenvalues are outside the unit circle, or
- if the eigenvalues on the unit circle are roots of unity?

6.3 Automatic Rational Sets

In Section 4.1 we showed how to compute the supremum, infimum and largest/smallest special points of the set

$$S = \left\{ \frac{a}{b} : (a, b) \in X \right\}$$

where $X \subseteq \mathbb{N}^2$ is a k-automatic set. Is this still possible X is an $\mathcal{N}$-automatic set for $\mathcal{N}$ some other numeration system? We conjecture that it is possible when $\mathcal{N}$ is the Fibonacci representation.

Recall that for k-automatic sets, we start with an automaton T that accepts X in base-k. Suppose we have an infinite walk $e_0e_1e_2\cdots$ in T, where $e_0e_1\cdots$ denote directed edges of T. Then we assign each e_i a weight, $w(e_i)$, proportional to the place value of the corresponding digit, and normalized so that the leading digit has weight 1. Since the places values in base-k are powers of k, we have $w(e_i) = kw(e_{i+1})$. We assign weights x_e to edges e of T by summing weights in our infinite path:

$$x_e = \sum_{i:e_i=e} w(e_i).$$

Since $w(e_i) = kw(e_{i+1})$ holds for all i, we see that the sum of the x_es going into some vertex is k times the sum of the x_es leaving the same vertex (ignoring the source at the

initial vertex). If we set up a linear program based on these conditions, it turns out that the optimal solutions correspond to infinite walks, and hence give us the supremum of S.

This works for Fibonacci representation until we get to the point where $w(e_i) = kw(e_{i+1})$. Instead, since the place values are Fibonacci numbers, we have $w(e_i) = w(e_{i+1}) + w(e_{i+2})$. This is not a condition that can be enforced at each vertex in the digraph of T. Instead, consider the *line digraph* L of T where, loosely speaking, each vertex of L corresponds to an edge in T, and each edge in L corresponds to a walk of length 2 in T. We can translate the infinite walk $e_0e_1e_2\cdots$ in the digraph of T to an infinite walk $(e_0, e_1)(e_1, e_2)(e_2, e_3)\cdots$ on L. The edge (e_i, e_{i+1}) in L has a pair of weights, $x_i = w(e_i)$ and $y_i = w(e_{i+1})$ associated with it. The next edge, (e_i, e_{i+1}) also has a pair of weights, $x_{i+1} = w(e_{i+1})$ and $y_{i+1} = w(e_{i+2})$. Hence, at any vertex in L we have $y_i = x_{i+1}$ (for consistency) and $x_i = y_i + y_{i+1}$ (the Fibonacci recurrence). We conjecture that a linear program based on L with a pair of weights (x_e, y_e) on each edge with the constraints

$$\sum_{e\in\text{in}(v)} y_e = \sum_{e\in\text{out}(v)} x_e$$
$$\sum_{e\in\text{in}(v)} x_e = \sum_{e\in\text{in}(v)} y_e + \sum_{e\in\text{out}(v)} y_e$$

will have optimal value $\sup S$.

Problem 6.4. Let $\mathcal{N}$ be the Fibonacci numeration system, and let $X \subseteq \mathbb{N}^2$ be an $\mathcal{N}$-automatic set. Is there a linear fractional program that computes

$$\sup\left\{\frac{a}{b} : (a, b) \in X\right\}$$

based on the line digraph construction suggested above?

And more generally,

Problem 6.5. Let $\mathcal{N}$ be a morphic numeration system. Is there an algorithm that computes

$$\sup\left\{\frac{a}{b} : (a, b) \in X\right\}$$

for a given $\mathcal{N}$-automatic set X?

6.4 Shift Orbit Closure

Recall that the shift orbit closure of a word $w \in \Gamma^\omega$ is a set $S \subseteq \Gamma^\omega$ such that for all $z \in S$ every prefix of z is a subword of w. We have seen a predicate for the nth symbol in the lexicographically least element of the shift orbit closure of a k-automatic sequence, and therefore the lexicographically least element is k-automatic and computable.

Suppose we are given a word $w \in \Gamma^\omega$ that is not recurrent. Rampersad [42] gives a non-constructive proof that S, the shift orbit closure of w, contains a recurrent word. This raises questions about the set of recurrent words in S.

Problem 6.6. Let $w \in \Gamma^\omega$ be a k-automatic word, and let $S \subseteq \Gamma^\omega$ be the orbit closure of w. Can we construct a k-automatic recurrent word in S? Is the lexicographically least recurrent word in S necessarily k-automatic?

One possible approach to this problem is based on the observation that, like the family of paperfolding words, the shift orbit closure of w is (usually) an uncountably infinite set of infinite words. Hence, like the paperfolding words, we may be able to extend our theory to quantify over elements of the shift orbit closure. There are three points that make the paperfolding extension possible. First, there is a sequence of instructions for each paperfolding sequence. Second, the first n instructions describe approximately 2^n symbols in the paperfolding word. Third, there is a DFAO that computes the ith symbol in a paperfolding word given the bits of i in parallel with an equal number of instructions.

Note that for the shift orbit closure, the first k^n symbols are some subword of w. We can specify any subword of length k^n by the position of its first occurrence. The first occurrence of any subword is less than $A_w(k^n)$, where $A_w(m)$ is the appearance function of w. Since w is k-automatic, the appearance function is k-synchronized and hence $A_w(m) = O(m)$ by Theorem 2.23. Hence, we can describe the first k^n symbols of an element of S with a string of digits (representing a position) of length $n + O(1)$. Furthermore, there is an automaton which, given an index i and position j (with $0 \leq i < k^n$ and $0 \leq j < A_w(k^n)$) in base-k, computes $w[i + j]$, the ith element of the subword specified by position j.

In summary, given an arbitrary element $z \in S$, we can describe $z[0..k^n - 1]$ by the position where it first occurs in w, which turns out to be $O(k^n)$. This description takes $n + O(1)$ digits, and there is an automaton for indexing into z, given a position $0 \leq i < k^n$ and the description. Unfortunately, the position of $z[0..k^n - 1]$ and the position of $z[0..k^{n+1} - 1]$ might be completely different, so it is not clear how to generate an infinite instruction sequence for z with the necessary properties, but it is certainly plausible that some related encoding will work.

Problem 6.7. Suppose $w \in \Gamma^\omega$ is a k-automatic word with shift orbit closure $S \subseteq \Gamma^\omega$. Is there an alphabet Σ and ω-language $L \subseteq \Sigma^\omega$ such that

- there is a bijection $f\colon L \to S$,
- L is regular, and
- there is an ω-automaton that takes a word $x \in L$ and a position i in base-k as input and computes $f(x)[i]$?

Can we extend our first-order theory to the shift orbit closure, similar to our extension for paperfolding words?

Note that if the answer is "yes", then we can immediately answer many of our earlier questions about recurrent words in S, since they can be expressed with predicates.

References

[1] J.-P. Allouche and M. Bousquet-Mélou. Facteurs des suites de Rudin-Shapiro généralisées. *Bull. Belg. Math. Soc.*, 1:145–164, 1994.

[2] Jean-Paul Allouche. The number of factors in a paperfolding sequence. *Bulletin of the Australian Mathematical Society*, 46:23–32, 1992.

[3] Jean-Paul Allouche and Mireille Bousquet-Mélou. Canonical positions for the factors in paperfolding sequences. *Theoretical Computer Science*, 129(2):263–278, 1994.

[4] Jean-Paul Allouche, Narad Rampersad, and Jeffrey Shallit. Periodicity, repetitions, and orbits of an automatic sequence. *Theor. Comput. Sci.*, 410(30–32):2795–2803, 2009.

[5] Jean-Paul Allouche and Jeffrey Shallit. The ring of k-regular sequences. *Theor. Comput. Sci.*, 98(2):163–197, 1992.

[6] Jean-Paul Allouche and Jeffrey O. Shallit. *Automatic Sequences : Theory, Applications, Generalizations*. Cambridge University Press, 2003.

[7] Yu Hin Au, Aaron Robertson, and Jeffrey Shallit. Van der Waerden's theorem and avoidability in words. *Integers*, 11:61–76, 2011.

[8] Vince Bárány. A hierarchy of automatic ω-words having a decidable MSO theory. *RAIRO — Theor. Inf. Appl.*, 42(3):417–450, 2008.

[9] Marie-Pierre Béal and Olivier Carton, editors. *Developments in Language Theory — 17th International Conference, DLT 2013, Marne-la-Vallée, France, June 18-21, 2013. Proceedings*, volume 7907 of *Lecture Notes in Computer Science*. Springer, 2013.

[10] V. Bruyère. Entiers et automates finis, mémoire de fin d'études. Master's thesis, University of Mons, Belgium, 1985.

[11] Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc*, 1:191–238, 1994.

[12] Julius R. Büchi. On a decision method in restricted second-order arithmetic. In *International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[13] Arturo Carpi and Cristiano Maggi. On synchronized sequences and their separators. *RAIRO — Theor. Inf. Appl.*, 35(6):513–524, 2001.

[14] Olivier Carton and Wolfgang Thomas. The monadic theory of morphic infinite words and generalizations. *Information and Computation*, 176(1):51–65, 2002.

[15] Julien Cassaigne, James D. Currie, Luke Schaeffer, and Jeffrey Shallit. Avoiding three consecutive blocks of the same size and same sum. *CoRR*, abs/1106.5204, 2011. To appear, *J. ACM*.

[16] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3–4):181–186, 1962.

[17] Alan Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical Systems Theory*, 3(2):186–192, 1969.

[18] Alan Cobham. Uniform tag sequences. *Mathematical Systems Theory*, 6(1-2):164–192, 1972.

[19] John H. Conway. *On Numbers and Games*. AK Peters, Ltd., 2nd edition, Dec 2000.

[20] Ethan M. Coven and G. A. Hedlund. Sequences with minimal block growth. *Mathematical Systems Theory*, 7(2):138–153, 1973.

[21] James D. Currie and Narad Rampersad. A proof of Dejean's conjecture. *Math. Comput.*, 80(274):1063–1070, 2011.

[22] James D. Currie and Narad Rampersad. Fixed points avoiding abelian k-powers. *J. Comb. Theory, Ser. A*, 119(5):942–948, 2012.

[23] Adrian Horia Dediu and Carlos Martín-Vide, editors. *Language and Automata Theory and Applications — 6th International Conference, LATA 2012, A Coruña, Spain, March 5–9, 2012. Proceedings*, volume 7183 of *Lecture Notes in Computer Science*. Springer, 2012.

[24] Adrian Horia Dediu, Carlos Martín-Vide, and Bianca Truthe, editors. *Language and Automata Theory and Applications — 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, volume 7810 of *Lecture Notes in Computer Science*. Springer, 2013.

[25] F. M. Dekking. Iteration of maps by an automaton. *Discrete Mathematics*, 126(1–3):81–86, 1994.

[26] Calvin C. Elgot and Michael O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *J. Symb. Log.*, 31(2):169–181, 1966.

[27] C. Frougny. Fibonacci representations and finite automata. *IEEE Trans. Inf. Theor.*, 37(2):393–399, March 1991.

[28] Christiane Frougny and Jacques Sakarovitch. Synchronized rational relations of finite and infinite words. *Theor. Comput. Sci.*, 108(1):45–82, 1993.

[29] Daniel Goč. Automatic theorem proving. Master's thesis, University of Waterloo, 2013.

[30] Daniel Goč, Luke Schaeffer, and Jeffrey Shallit. Subword complexity and k-synchronization. In Béal and Carton [9], pages 252–263.

[31] Daniel Goč and Jeffrey Shallit. Primitive words and Lyndon words in automatic sequences. *CoRR*, abs/1207.5124, 2012.

[32] Štěpán Holub. Abelian powers in paper-folding words. *J. Comb. Theory, Ser. A*, 120(4):872–881, 2013.

[33] Veikko Keranën. Abelian squares are avoidable on 4 letters. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 41–52. Springer Berlin Heidelberg, 1992.

[34] C. Kimberling. The Zeckendorf array equals the Wythoff array. *Fibonacci Quart.*, 33:3–8, 1995.

[35] P. B. A. Lecomte and M. Rigo. Numeration systems on a regular language. *Theory of Computing Systems*, 34(1):27–44, 2000.

[36] J. A. Leech. A problem on strings of beads. *Math. Gaz.*, 41:277–278, 1957.

[37] Arnaud Maes. An automata theoretic decidability proof for first-order theory of $\langle \mathbb{N}, <, p \rangle$ with morphic predicate p. *Journal of Automata, Languages and Combinatorics*, 4(3):229–246, 1999.

[38] Marston Morse and Gustav A. Hedlund. Symbolic dynamics. *American Journal of Mathematics*, 60(4):815–866, Oct 1938.

[39] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, October 1966.

[40] Dominique Perrin and Jean-Eric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.

[41] Mojzesz Presburger and Dale 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.

[42] Narad Rampersad. Non-constructive methods for avoiding repetitions in words, 2013. To appear at WORDS 2013.

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

[44] Gwénaël Richomme, Kalle Saari, and Luca Q. Zamboni. Abelian complexity of minimal subshifts. *Journal of the London Mathematical Society*, 2010.

[45] Michel Rigo. Generalization of automatic sequences for numeration systems on a regular language. *Theor. Comput. Sci.*, 244(1-2):271–281, 2000.

[46] Eric Rowland and Jeffrey Shallit. k-automatic sets of rational numbers. In Dediu and Martín-Vide [23], pages 490–501.

[47] Kalle Saari. *On the frequency and periodicity of infinite words*. PhD thesis, University of Turku, 2008.

[48] Luke Schaeffer. Ostrowski numeration and the local period of sturmian words. In Dediu et al. [24], pages 493–503.

[49] Luke Schaeffer and Jeffrey Shallit. The critical exponent is computable for automatic sequences. *Int. J. Found. Comput. Sci.*, 23(8):1611–1626, 2012.

[50] Alfred Tarski, Andrzej Mostowski, and Raphael M. Robinson. *Undecidable Theories*. North-Holland, 1953.