Tomi Kärki

Automatic Sequences and their Shuffles

Turku Centre *for* Computer Science

TUCS Technical Report
No 745, February 2006

Automatic Sequences and their Shuffles

Tomi Kärki
Department of Mathematics
University of Turku
FIN-20014 Turku, Finland
topeka@utu.fi

TUCS Technical Report
No 745, February 2006

Abstract

Infinite sequences are basic objects in modern discrete mathematics and theoretical computer science. Especially, sequences generated by some simple model of computation, for example by morphisms or automata, are widely studied. This work focuses on automatic sequences. Such a sequence can be generated by a finite deterministic automaton with output. From another viewpoint, automatic sequences are fixed points of uniform morphisms under a coding.

In this work we present four equivalent definitions of automatic sequences using finite automata with output, kernels, fibers and uniform morphisms. Basic closure properties of this class of sequences are considered. We also mention how automatic sequences are related to transcendental number theory via the subword complexity of a sequence. In addition we give a detailed proof of the famous Cobham's theorem. It states that if a set is both k- and l-automatic, for two multiplicatively independent integers k and l, then it is ultimately periodic. Particularly, sequences are consider from an algorithmic point of view in context with so called regular shuffles. We present algorithms, which enable us to easily calculate generating morphisms and codings for sequences obtained by shuffling fixed points of uniform morphisms.

Keywords: automatic sequences, finite automata with output, Cobham's theorem, regular shuffles, perfect shuffles

TUCS Laboratory
Discrete Mathematics for Information Technology

Contents

Preface **1**

1 Automata and regular languages **3**
1.1 Definitions . . . 3
1.2 Closure properties and pumping lemma . . . 6
1.3 Finite automata with output . . . 9
1.4 Minimization . . . 12

2 Automatic sequences **19**
2.1 Definition and examples . . . 19
2.2 Fibers . . . 22
2.3 Kernels . . . 23
2.4 Uniform morphisms . . . 25
2.5 Closure properties . . . 27
2.6 Subword complexity . . . 31

3 Cobham's theorem **35**
3.1 Multiplicative independence . . . 35
3.2 Automatic sets . . . 37
3.3 Right dense and syndetic sets . . . 38
3.4 Equivalence relations . . . 42
3.5 Proof of Cobham's theorem . . . 43

4 Shuffles of automatic sequences **49**
4.1 Definitions . . . 49
4.2 Perfect shuffles . . . 50
4.2.1 Perfect k-shuffle of k-automatic sequences . . . 50
4.2.2 Perfect m-shuffle of k-automatic sequences . . . 54
4.3 Regular shuffles . . . 58
4.4 Applying algorithms to Schröder numbers . . . 64
4.5 On complexities of the algorithms . . . 68
4.6 Final Remarks . . . 70

References **72**

Index **75**

Preface

Numbers and sequences of numbers conceal numerous mysteries which have intrigued people from ancient times. The golden ratio $\frac{1+\sqrt{5}}{2} = 1,618033\ldots$, $\pi = 3,14159265\ldots$, the sequence of primes $2, 3, 5, 7, 11, 13, 17, 19, \ldots$ and the Thue-Morse sequence $0, 1, 1, 0, 1, 0, 0, 1, 1, 0, \ldots$ are examples of mathematical objects which appear in many surprising contexts from nature to art and science. Well known are also the puzzles where a logical continuation of a given sequence should be found. For example, how does this sequence $2, 12, 1112, 3112, 132112, 311322, \ldots$ continue? Consider the number of occurrences of different digits of the previous term or consult Sloane's web page `http://www.research.att.com/~njas/sequences/index.html`, which contains over 100000 sequences. They are not just for fun. Sequences are widely studied in the area of modern mathematics and theoretical computer science. There is even a scientific web journal, *Journal of Integer Sequences*, and a periodic international conference, *Sequences and Their Applications*, devoted to this topic.

One of the main characteristics of sequences studied in mathematics is their randomness. Some sequences are highly ordered, even periodic, while others have no simple description. This concept of order or, from the opposite point of view, the concept of complexity of a sequence has connections to transcendental number theory. For example, all decimal expansions of irrational algebraic numbers are conjectured to be normal, i.e. the decimal expansion contains every block of digits of length n with a frequency asymptotic to $1/10^n$. This conjecture is based on the fact that algebraic irrational numbers are badly approximable by rational numbers. Thus number theory gives good motivation for the study of sequences.

This work focuses on *automatic sequences*. They form a class of sequences between simple order and complex disorder. The presentation is mainly based on the first integrated treatment of these sequences, the recently published book [4] by J.-P. Allouche and J. Shallit. In this context, we are especially interested in constituting new automatic sequences by shuffling given automatic sequences in a regular way. Algorithmic aspects of these shuffles are emphasized.

The text is organized as follows. In Section 1 we introduce our notation and recall some basic results on automata and formal language theory. A good and detailed reference for these results is, for example, the book of S. Eilenberg [10]. In Section 2 we first define automatic sequences using finite automata with output and then we give three other characterizations in terms of fibers, kernels and uniform morphisms. Basic closure properties are also considered in this section. Section 3 is devoted to a famous result concerning automatic sequences, known as Cobham's Theorem. It characterizes sequences which are automatic in two multiplicatively independent bases k and l. We give a detailed proof of this result. Finally, Section 4 deals with shuffles. Perfect shuffles and regular shuffles are treated in their own subsections. Our main goal is to introduce simple algorithms for calculation of generating morphisms and codings of these shuffles. The algorithms are based on a connection between automata and morphisms. Com-

plexities of the algorithms are also briefly studied. In the end we consider some generalizations.

Acknowledgements. I am grateful to my supervisors, professor Juhani Karhumäki and professor Tero Harju, for all the support and advise during the course of this work. I would also like to thank doctor Ari Renvall for useful discussions and comments. Special thanks are due to professor Jean Berstel, doctor Julien Cassaigne and doctor Gwénaël Richomme for comments and suggestions during my visit in France. I also thank professor M. Rigo and docent J. Honkala for many valuable comments that made the exposition more readable.

1 Automata and regular languages

In this section we introduce the notation and recall some basic results on automata theory. We begin by considering finite automata, regular languages and their connection to rational languages. The second subsection is devoted to different closure properties of regular languages. In order to define automatic sequences we introduce finite automata with output in the third subsection. Finally, minimization of an automaton is considered.

1.1 Definitions

We use the following notation: $\mathbb{N} = \{0, 1, 2, \ldots\}$. The set of all functions from a set A to a set B is denoted by $B^A = \{f \mid f\colon A \to B\}$.

We start with some basic definitions from language theory. An *alphabet* Σ is a nonempty finite set of symbols, called *letters*, and a *word* over Σ is a (finite or infinite) sequence of symbols from Σ. The empty word is denoted by ε. Denote by Σ^* and Σ^+ the sets of all finite words and finite nonempty words over Σ, respectively. A *catenation* of finite words is an operation defined by $a_1 \cdots a_n \cdot b_1 \cdots b_m = a_1 \cdots a_n b_1 \cdots b_m$ for $a_i, b_i \in \Sigma$. The *reversal* of a word $w = a_0 a_1 \cdots a_{n-1} a_n$ is $w^R = a_n a_{n-1} \cdots a_1 a_0$. The *length* of a word w, denoted by $|w|$, is the total number of (occurrences of) letters in w. The number of letters a in w is denoted by $|w|_a$. A word w is a *factor* of a word u (resp. a left factor or a *prefix*, a right factor or a *suffix*), if there exist words x and y such that $u = xwy$ (resp. $u = wy, u = xw$). Denote the set of all factors of w of length n by $L_n(w)$. By $w(n)$ we mean the $(n+1)$th letter of the word $w \in \Sigma^*$, where $n = 0, 1, \ldots, |w| - 1$. Sometimes we use the notation w_n instead of $w(n)$. Finite words are usually denoted by u, v or w and infinite words are denoted by bold letters.

For an infinite *sequence* $u_0 u_1 u_2 \cdots$ we use the notation $\mathbf{u} = (u_n)_{n \geq 0}$, where $u_n = \mathbf{u}(n)$ is the $(n+1)$th symbol of $\mathbf{u}$. Then $\mathbf{u}$ can be regarded as a mapping from $\mathbb{N}$ to Σ. A sequence is called *ultimately periodic* if it is of the form $xy^\omega = xyyy\cdots$ for some $x \in \Sigma^*$ and for some $y \in \Sigma^+$. A special sequence related to a subset S of $\mathbb{N}$ is the *characteristic sequence* $(\chi_S(n))_{n \geq 0}$ of S. It is defined by the *characteristic function*

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

A *language* is a subset of Σ^*. For languages $L, K \subseteq \Sigma^*$ and for a word $u \in \Sigma^*$ we define

$$\begin{aligned} L \setminus K &= \{w \mid w \in L, w \notin K\}, \\ LK &= \{uv \mid u \in L, v \in K\}, \\ L^R &= \{w^R \mid w \in L\}, \end{aligned}$$

$$
\begin{aligned}
L^* &= \bigcup_{i\geq 0} L^i = \{u_1u_2\cdots u_n \mid n \geq 0, u_i \in L\},\\
L^+ &= \bigcup_{i\geq 1} L^i = \{u_1u_2\cdots u_n \mid n \geq 1, u_i \in L\},\\
u^{-1}K &= \{v \mid uv \in K\},\\
Ku^{-1} &= \{v \mid vu \in K\},\\
L^{-1}K &= \bigcup_{u\in L} u^{-1}K = \{v \mid \exists\, u \in L\colon uv \in K\}.
\end{aligned}
$$

A *morphism* is a mapping $\varphi\colon \Sigma^* \to \Delta^*$ such that

$$\varphi(vw) = \varphi(v)\varphi(w)$$

for any two words $v, w \in \Sigma^*$. It is determined by the images of the letters $a \in \Sigma$. If $\varphi(a) \neq \varepsilon$ for all $a \in \Sigma$, then φ is *nonerasing*. A morphism φ is called *k-uniform* or *of constant length k,* if $|\varphi(a)| = k$ for every $a \in \Sigma$. A 1-uniform morphism is also called a *coding*.

One of the most basic models of computation (by finite memory device) is a *deterministic finite automaton*, DFA. Formally, it is a quintuple

$$M = (Q,\, \Sigma,\, \delta,\, q_0,\, F),$$

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

The domain of δ is extended as follows: Define $\delta(q, \varepsilon) = q$ for all $q \in Q$ and $\delta(q, xa) = \delta(\delta(q, x), a)$ for all $q \in Q$, $x \in \Sigma^*$, and $a \in \Sigma$. Then $L(M)$, the language *accepted* by M, is defined to be

$$L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}.$$

If the input alphabet is $\Sigma_k = \{0, 1, \ldots, k-1\}$, then the automaton is called a k-DFA. The languages accepted by DFAs are called *recognizable* or *regular* languages. In Figure 1 we illustrate a deterministic finite automaton using a labeled *transition graph.* An edge $p \xrightarrow{a} q$ corresponds to a transition $\delta(p, a) = q$. The initial state is denoted by a small incoming arrow and the accepting states are marked with double circles. The language accepted by the DFA is easily seen to consist of the binary words with suffix ab.

Sometimes it is convenient to allow zero or more than one distinct transitions on a state-input pair. It means that transition functions can be considered as functions from $Q \times \Sigma$ into the set of all subsets of Q. Equivalently, we may replace functions with relations. In other words, we define a *nondeterministic finite automaton,* NFA, by a quintable $M = (Q,\, \Sigma,\, E,\, Q_0,\, F)$, where $E \subseteq Q \times \Sigma \times Q$ is

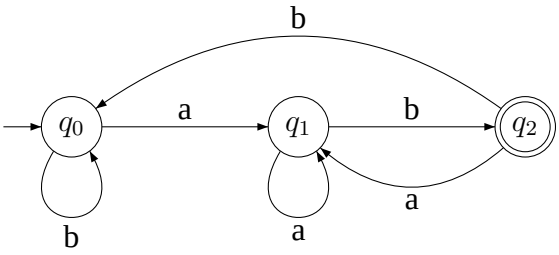


Figure 1: DFA accepting words with suffix ab

a *transition relation* and $Q_0 \subseteq Q$ is a set of initial states. The language accepted by an NFA M is

$$\begin{aligned} L(M) &= \{w = a_1 \cdots a_n \in \Sigma^* \mid \exists\, q_0, \ldots, q_n \in Q\colon\, q_0 \in Q_0, q_n \in F, \\ &\quad (q_{i-1}, a_i, q_i) \in E \text{ for } i = 1, \ldots, n\}. \end{aligned}$$

In a transition graph of a nondeterministic automaton there exists an edge $p \xrightarrow{a} q$ if (p, a, q) belongs to E. An example of such a graph is given in Figure 2.

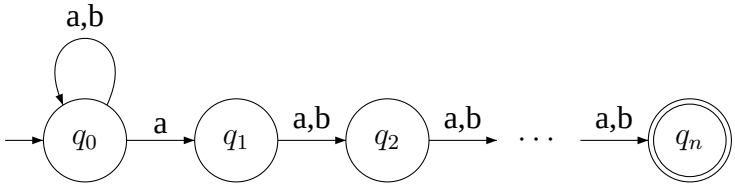


Figure 2: Transition graph of an NFA

A connection between NFAs and DFAs can be formulated as follows.

Theorem 1. *If L is accepted by an NFA with n states, then L is accepted by a DFA with at most 2^n states.*

The proof is based on the so called *subset construction.* Suppose that L is accepted by an NFA $A = (Q, \Sigma, E, Q_0, F)$. Then L is accepted by a DFA $B = (2^Q, \Sigma, \delta, q_0, G)$, where

$$\begin{aligned} \delta(S, a) &= \{q \in Q \mid \exists p \in S\colon (p, a, q) \in E\} \quad \text{for } S \in 2^Q \text{ and } a \in \Sigma, \\ q_0 &= \{Q_0\}, \\ G &= \{H \in 2^Q \mid H \cap F \neq \emptyset\}. \end{aligned}$$

It can be shown that the exponential growth in the number of states when transferring an NFA to a DFA cannot be avoided in general. For example, consider the deterministic automata corresponding to the NFAs of Figure 2 accepting the languages $L_n = \{wav \mid |v| = n-1\}$ for $n \geq 1$, or see [21].

Next we define *rational expressions* over an alphabet Σ as follows:

(i)	$\emptyset$ and a are rational expressions for each $a \in \Sigma$.
(ii-iv)	If α and β are rational expressions, so are $(\alpha + \beta)$, $(\alpha\beta)$ and $(\alpha)^*$.
(v)	These are all rational expressions.

A language $L = L(u)$ specified by a rational expression u is defined naturally and it is called *rational*. This means that

(i)	the empty set $\emptyset$ and the singleton sets $\{a\}$ are rational languages,
(ii)	$L(\alpha + \beta) = L(\alpha) \cup L(\beta)$,
(iii)	$L(\alpha\beta) = L(\alpha)L(\beta) = \{rs \mid r \in L(\alpha), s \in L(\beta)\}$,
(iv)	$L(\alpha^*) = L(\alpha)^* = \{r_1 \cdots r_n \mid n \geq 0, r_i \in L(\alpha)\}$.

The celebrated result by Kleene connects rational and regular languages.

Theorem 2. (Kleene) *A language $L \subseteq \Sigma^*$ is regular if and only if it is rational.*

A proof of the theorem can be found, for example, in the book of Eilenberg [10].

1.2 Closure properties and pumping lemma

Next we summarize some basic properties of regular languages. The class of regular languages is closed under Boolean operations.

Theorem 3. *Let $L, L_1, L_2 \subseteq \Sigma^*$ be regular languages. Then also $\Sigma^* \setminus L$, $L_1 \cap L_2$ and $L_1 \cup L_2$ are regular.*

The class is also closed under rational operations.

Theorem 4. *Let $L, L_1, L_2 \subseteq \Sigma^*$ be regular languages. Then also L_1L_2 and L^* are regular.*

With respect to morphisms we have:

Theorem 5. *Let $L \subseteq \Sigma^*$ and $M \subseteq \Delta^*$ be regular languages and $\varphi\colon \Sigma^* \to \Delta^*$ be a morphism. Then $\varphi(L)$ and $\varphi^{-1}(M) = \{w \in \Sigma^* \mid \varphi(x) \in M\}$ are regular.*

Also reversing is considered.

Theorem 6. *If L is a regular language, then so is L^R.*

The next three closure properties are related to so called k-*automatic sets* of natural numbers; see Section 3.2. We consider representations of nonnegative integers in an integer base $k \geq 2$. For any nonnegative integer n there exists a unique representation $(n)_k = w_0 w_1 \cdots w_r$ such that

$$n = \sum_{i=0}^{r} w_i k^{r-i}, \tag{1}$$

where $w_i \in \Sigma_k = \{0, 1, \ldots, k-1\}$ for $i = 0, 1, \ldots, r$ and $w_0 \neq 0$. Note that $(0)_k = \varepsilon$. This *normalized representation* $(n)_k$ is a word in the language $C_k = \{\varepsilon\} \cup (\Sigma_k \setminus \{0\})\,\Sigma_k^*$, i.e. there are no zeros in the beginning of the word. We may also define an inverse operation as follows. For any word in Σ_k^* we define

$$[w_0 w_1 \cdots w_r]_k = \sum_{i=0}^{r} w_i k^{r-i}. \tag{2}$$

Note that leading zeros are allowed in (2). Clearly we have $[(n)_k]_k = n$. For a set $S \subseteq \mathbb{N}$, we define $(S)_k = \{(s)_k \mid s \in S\}$, and $[L]_k = \{[w]_k \mid w \in L\}$ for any language $L \subseteq \Sigma_k^*$. Our next theorem deals with addition. There are some simpler proofs of this theorem, but the construction of the automaton introduced in the following proof will be useful in the sequel. Similar constructions can be found in the literature; see e.g. [29, Lemma 1].

Theorem 7. *Let $S \subseteq \mathbb{N}$. If $\{(n)_k \mid n \in S\}$ is a regular language, then so is $\{(n+1)_k \mid n \in S\}$.*

Proof. Suppose that $M = (Q, \Sigma_k, \delta, q_0, F)$ is an automaton accepting the language $L = \{(n)_k \mid n \in S\}$. Our aim is to construct an automaton M' accepting $L' = \{(n+1)_k \mid n \in S\}$. With input $(n+1)_k$ this automaton simulates the behavior of the automaton M with input $(n)_k$. Let us first consider these inputs. We divide the representations of the numbers $n+1$ into two cases. If $(n+1)_k = 10^p$, then $(n)_k = (k-1)^p$ and these two representations contain a different number of letters. Suppose now that $(n+1)_k = w_0 w_1 \cdots w_r$ is not of that form. We define an index $s = \max\{i \mid w_i \neq 0, 0 \leq i \leq r\}$. Then the representation of n is $w_0' w_1' \cdots w_r'$, where $w_i' = w_i$ for $i = 0, 1, \ldots, s-1$, $w_s' = w_s - 1$ and digits after w_s' are all $k-1$. It is essential for the correct behavior of the automaton M' to know the first digit starting from left in which the representations of integers $n+1$ and n differ. This is the digit w_s in our notation. But when reading the digits from left to right one cannot know "on-line" which digit is the last nonzero digit. One solution to this problem is to construct an automaton which guesses the changed digit w_s while reading the input and which is capable of making a new guess if the first guess proves to be false.

Our construction of $M' = (Q', \Sigma_k, \delta', q_0', F')$ is the following. We define $Q' = (Q \times Q) \cup \{q_0', g\}$, where q_0' is an initial state and g is a garbage state. Consider first the state pairs of the cartesian product $Q \times Q$. The first coordinate corresponds to the simulation of M before the changed digit. The second

coordinate corresponds to the situation, where the change has already happened. Thus the accepting states of M' are the states where the second coordinate is an accepting state of M. We have $F' = Q \times F$. The simulation is implemented in the transition function. For $(x, y) \in Q \times Q$ and $a \in \Sigma_k$, we have

$$\delta'((x,y),a) = \begin{cases} (\delta(x,a), \delta(y,k-1)) & \text{if } a = 0, \\ (\delta(x,a), \delta(x,a-1)) & \text{otherwise.} \end{cases}$$

Note that the transition in the case $a > 0$ depends only on the first coordinate. The second coordinate can be a right choice only if the rest of the input consists of zeros. The initial state must also deal with the case where the lengths of $(n)_k$ and $(n-1)_k$ are different. Thus, for $a \in \Sigma_k$, we define

$$\delta'(q_0', a) = \begin{cases} g & \text{if } a = 0, \\ (\delta(q_0, 1), q_0) & \text{if } a = 1, \\ (\delta(q_0, a), \delta(q_0, a-1)) & \text{otherwise.} \end{cases}$$

For the garbage state, define $\delta'(g, a) = g$ for every $a \in \Sigma_k$.

Next we show that this construction is correct. Consider first the representation of 0, which is ε. Since $\delta(q_0', \varepsilon) = q_0' \notin F'$, the automaton rejects the input, which is the correct action. Also representations with leading zeros are rejected with the help of the garbage state. With the input $(n)_k = 10^p$ our automaton ends up to the state $(\delta(q_0, 10^p), \delta(q_0, (k-1)^p))$. This is an accepting state if and only if $\delta(q_0, (k-1)^p)) \in F$, i.e. $n \in S$. Similar reasoning also applies to the input $a0^p$, where $1 < a \leq k-1$. In this case the final state is $(\delta(q_0, a0^p)$, $\delta(q_0, (a-1)(k-1)^p))$. Finally, consider the case where $|(m)_k| = |(m-1)_k|$, $(n+1)_k = w_0 \cdots w_r$ and w_s, $s > 0$, is defined as above. Let $\delta(q_0, w_0 \cdots w_{s-1}) = a$, $\delta(a, w_s - 1) = b$ and $\delta(b, (k-1)^{r-s}) = c$. If $r - s = 0$, then $(k-1)^{r-s}$ means the empty word. We have

$$\begin{aligned} \delta'(q_0', w_0 \cdots w_r) &= \delta'((\delta(q_0, w_0 \cdots w_{s-1}), y), w_s 0^{r-s}) \quad \text{(for some } y \in Q\text{)} \\ &= \delta'((a, y), w_s 0^{r-s}) \\ &= \delta'((\delta(a, w_s), \delta(a, w_s - 1)), 0^{r-s}) \\ &= \delta'((\delta(a, w_s), b), 0^{r-s}) \\ &= (x, \delta(b, (k-1)^{r-s})) \quad \text{(for some } x \in Q\text{)} \\ &= (x, c). \end{aligned}$$

This is an accepting state if and only if $\delta(q_0, (n)_k) = c \in F$. Thus $(n+1)_k$ is accepted if and only if $n \in S$. □

Corollary 1. *For all integers $c \geq 0$, if $\{(n)_k \mid n \in S\}$ is a regular language, then so is $\{(n+c)_k \mid n \in S\}$.*

Theorem 8. *Let $S \subseteq \mathbb{N}$. If $\{(n)_k \mid n \in S\}$ is a regular language, then so is $\{(bn)_k \mid n \in S\}$ for any integer $b \geq 0$.*

Proof. Suppose that $M = (Q, \Sigma_k, \delta, q_0, F)$ is an automaton accepting the language $L = \{(n)_k \mid n \in S\}$. Consider a k-DFA $M' = (Q', \Sigma_k, \delta', (q_0, 0), F')$, where

$$\begin{aligned} Q' &= Q \times \{0, 1, \ldots, b-1\}, \\ \delta'((q,j),l) &= (\delta(q, \lfloor (kj+l)/b \rfloor), kj + l \,(\mathrm{mod}\, b)), \\ F' &= \{(q,0) \in Q' \mid q \in F\}. \end{aligned}$$

This construction uses the school algorithm of division by b in the base k. Consider two states (q, j) and (q', j') in Q' such that $\delta'((q, j), l) = (q', j')$ for a letter $l \in \Sigma_k$. The second component j of the state $(q, j) \in Q'$ corresponds to the current remainder modulo b. In school algorithm we obtain the next digit of the quotient by reading the new digit l of the dividend and calculating $\lfloor (kj + l)/b \rfloor$. The new remainder j' is $kj + l \mod b$, which is by definition the second component of (q', j'). The change of the first component corresponds to the transition in M from q to the state q', when the digit $\lfloor (kj + l)/m \rfloor$ is read. Let $(N)_k$ be the input of M'. Thus in the first components of Q' we simulate the original automaton M with input $(\lfloor N/b \rfloor)_k$. If $N = bn + i$, then the first component of the final state is the final state of M with input $(n)_k$, i.e. the state $\delta(q_0, (n)_k)$, and the second component is i. Now it is clear that $\delta'((q_0, 0), (N)_k) \in F'$ if and only if $\delta(q_0, (n)_k) \in F$ and $i = 0$. Thus the accepted language is $\{(bn)_k \mid n \in S\}$. □

As a last general property of regular languages we recall the pumping lemma, which can be used, for example, to prove that some languages are not regular; cf. [4].

Lemma 1. *Let $L \subseteq \Sigma^*$ be a regular language. Then there exists a constant $n \geq 1$ such that for all strings $z \in L$ with $|z| \geq n$, there exists a decomposition $z = uvw$, where $u, v, w \in \Sigma^*$ and $|uv| \leq n$ and $|v| \geq 1$, such that $uv^i w \in L$ for all $i \geq 0$. Furthermore, the constant n can be taken to be the minimal number of states in an NFA accepting L.*

1.3 Finite automata with output

We are now going to generalize DFAs. Instead of thinking that a DFA either accepts or rejects an input string, we could consider it as a function $f\colon \Sigma^* \to \{0, 1\}$, where 1 represents acceptance and 0 rejection. Next we define a more general function $f'\colon \Sigma^* \to \Delta$ using automata by attaching an output symbol in Δ to each state. Thus, if the automaton enters a state q after reading the input, the output is $\tau(q)$, where $\tau\colon Q \to \Delta$ is the so called *output function*. Using previous notation a *deterministic automaton with output*, DFAO, is formally defined to be a 6-tuple

$$M = (Q, \Sigma, \delta, q_0, \Delta, \tau).$$

If the input alphabet is $\Sigma_k = \{0, 1, \ldots, k-1\}$, then the automaton is called a k-DFAO. The automaton M defines a function $f_M \colon \Sigma^* \to \Delta$ by the rule

$$f_M(w) = \tau(\delta(q_0, w)).$$

A function, which can be computed this way, is called a *finite-state function*. Transition graphs can be used to represent DFAOs in a similar way as they were used for DFAs; see Figure 3.

Example 1. Consider a 2-DFAO which with input $(n)_2$ computes the residue modulo 3 of the nonnegative integers n. The states of the automaton correspond to the residue classes and a state q is labeled by $q/\tau(q)$, where $\tau(q)$ is the residue class of the state. Transitions are defined by $\delta(q_i, a) = q_j$, where $j = 2i + a \pmod 3$. For example, when $w = 1010 = (10)_2$, then the last state is q_1 and the output is 1.

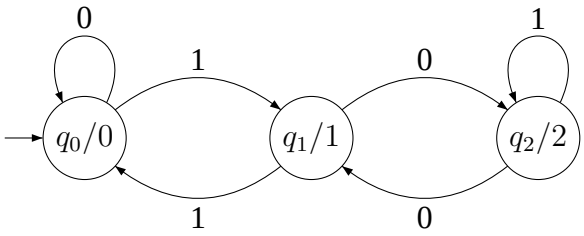


Figure 3: 2-DFAO computing the residue mod 3

Finite automata with output are closely related to DFAs and to regular languages.

Theorem 9. *Let $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ be a DFAO. Then for all $d \in \Delta$, the set $I_d(M) = \{w \in \Sigma^* \mid \tau(\delta(q_0, w)) = d\}$ is a regular language.*

Proof. Define a DFA $M_d = (Q, \Sigma, \delta, q_0, F_d)$ with $F_d = \{q \in Q \mid \tau(q) = d\}$. Clearly, the corresponding regular language $L(M_d)$ is $I_d(M)$. □

The sets $I_d(M)$ are called *fibers*. Next we prove how DFAOs can be constructed by "gluing together" DFAs for different regular languages.

Theorem 10. *Let Σ be an alphabet and let $L_1, L_2, \ldots, L_r$ be regular languages that partition Σ^*. Let $\Delta = \{a_1, a_2, \ldots, a_r\}$ be a new alphabet. Then there exists a DFAO $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ such that $L_i = I_{a_i}(M)$.*

Proof. Since each L_i is regular, there exists a DFA $M_i = (Q_i, \Sigma, \delta_i, q_{0i}, F_i)$ accepting L_i. Define a DFAO $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ as follows:

$$\begin{aligned} Q &= Q_1 \times Q_2 \times \cdots \times Q_r, \\ q_0 &= (q_{01}, q_{02}, \ldots, q_{0r}), \\ \delta((q_1, q_2, \ldots, q_r), a) &= (\delta_1(q_1, a), \delta_2(q_2, a), \ldots, \delta_r(q_r, a)) \text{ and} \\ \tau((q_1, q_2, \ldots, q_r)) &= \begin{cases} a_j & \text{if } q_j \in F_j \text{ for exactly one } j, \\ a_1 & \text{otherwise.} \end{cases} \end{aligned}$$

Suppose first that w belongs to L_j. Then the jth coordinate in the state $\delta(q_0, w)$ of M is $\delta_j(q_{0j}, w) \in F_j$. Since the sets L_i partition Σ^*, other coordinates of $\delta(q_0, w)$ are not final. Thus $\tau(\delta(q_0, w)) = a_j$. Secondly, if $\tau(\delta(q_0, w)) = a_j$, then, by definition of τ, we have $\delta_j(q_{0j}, w) \in F_j$. Thus $w \in L_j$. □

Next we consider finite-state functions and reversed inputs.

Theorem 11. *Let $f\colon \Sigma^* \to \Delta$ be a finite-state function computable by a DFAO with n states. Then f^R defined by $f^R(w) = f(w^R)$ is a finite-state function computable by a DFAO with at most $|\Delta|^n$ states.*

Proof. Suppose that $f(w) = \tau(\delta(q_0, w))$ for a DFAO $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$, where $|Q| = n$. We construct a DFAO

$$M' = (Q', \Sigma, \delta', q_0', \Delta, \tau'),$$

such that $\tau'(\delta'(q_0', w)) = f^R(w)$, i.e. $\tau'(\delta'(q_0', w)) = f(w^R) = \tau(\delta(q_0, w^R))$. There is a finite number, more precisely $|\Delta|^n$, functions from Q to Δ. Especially, for each $w \in \Sigma^*$, let $h_w\colon Q \to \Delta$ be defined by

$$h_w(q) = \tau(\delta(q, w^R)). \qquad (*)$$

Our goal is to construct an automaton with the set of states $Q' = \Delta^Q = \{f \mid f\colon Q \to \Delta\}$ such that $\delta'(q_0', w) = h_w$ for every $w \in \Sigma^*$. Then defining the output function by the rule

$$\tau'(g) = g(q_0)$$

for all $g \in Q'$, we have $\tau'(\delta'(q_0', w)) = \tau'(h_w) = h_w(q_0) = \tau(\delta(q_0, w^R))$.

The initial state of the automaton M' must be $q_0' = \delta'(q_0', \varepsilon) = h_\varepsilon = \tau$, since $h_\varepsilon(q) = \tau(\delta(q, \varepsilon)) = \tau(q)$ for all $q \in Q$. Let us next consider the transition function δ'. We define

$$\delta'(g, a) = h, \text{ where } h(q) = g(\delta(q, a)).$$

Now it suffices to prove that $\delta'(q_0', w) = h_w$ for every word $w \in \Sigma^*$. We prove it by induction on $|w|$. First, if $|w| = 0$, i.e. $w = \varepsilon$, the case is clear. Secondly, suppose that the claim holds for $|w| = n$. We proceed to $|w| = n + 1$.

Let $w = xa$, where $|x| = n$ and $a \in \Sigma$. Then $\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a) = \delta'(g, a) = h$, where by the induction hypothesis $g(q) = h_x(q) = \tau(\delta(q, x^R))$ for every $q \in Q$. Then $h(q) = g(\delta(q, a)) = \tau(\delta(\delta(q, a), x^R)) = \tau(\delta(q, ax^R)) = \tau(\delta(q, w^R))$. This completes the induction. The claim concerning the number of states in M' follows from the construction. □

Note that as a corollary of Theorem 11 we obtain Theorem 6 by considering the output alphabet $\Delta = \{0, 1\}$.

1.4 Minimization

In this subsection we consider minimization of DFAs and, as a generalization, minimization of deterministic automata with output. We say that a DFA M is *minimal* if it has the smallest number of states among all DFAs M' with $L(M') = L(M)$. Minimization is closely related to the so called Myhill-Nerode theorem, which gives another characterization of regular languages. In order to present the theorem we first need to consider some equivalence relations.

Recall that an equivalence relation $\sim$ on Σ^* is a relation, which is reflexive, symmetric and transitive. This means that for $u, v, w \in \Sigma^*$

$$\begin{aligned} u &\sim u, \\ u \sim v &\implies v \sim u, \\ u \sim v,\ v \sim w &\implies u \sim w. \end{aligned}$$

An equivalence relation partitions Σ^* into a number of disjoint equivalence classes. By $[w]$ we mean the equivalence class containing the word w. The word w is called a *representative* of the class. If the number of the equivalence classes is finite, the relations is said to be of *finite index.* An equivalence relation is said to be *right-invariant* if for any $u, v, w \in \Sigma^*$ we have

$$u \sim v \implies uw \sim vw.$$

Such a relation is also called a *right congruence.* A particularly important equivalence relation related to an arbitrary language $L \subseteq \Sigma^*$, called *Myhill-Nerode equivalence,* is defined by

$$u \sim_L v \iff u^{-1}L = v^{-1}L.$$

It is easily seen to be right-invariant:

$$\begin{aligned} u \sim_L v &\implies u^{-1}L = v^{-1}L \\ &\implies w^{-1}(u^{-1}L) = w^{-1}(v^{-1}L) \\ &\implies (uw)^{-1}L = (vw)^{-1}L \\ &\implies uw \sim_L vw. \end{aligned}$$

Another equivalence relation related to the regular language $L(M)$, via the DFA $M = (Q, \Sigma, \delta, q_0, F)$, is defined by

$$u \sim_M v \iff \delta(q_0, u) = \delta(q_0, v).$$

Since $\delta(q_0, uw) = \delta(\delta(q_0, u), w) = \delta(\delta(q_0, v), w) = \delta(q_0, vw)$ for equivalent words u, v and for all $w \in \Sigma^*$, also $\sim_M$ is right invariant. Actually, for a regular language $L = L(M)$ the relation $\sim_M$ is a *refinement* of $\sim_L$, i.e. the equivalence classes of $\sim_L$ are unions of equivalence classes of $\sim_M$ as stated in the following lemma.

Lemma 2. *Let $L = L(M)$ for a DFA $M = (Q, \Sigma_k, \delta, q_0, F)$. Then for all $u, v \in \Sigma^*$:*

$$u \sim_M v \quad \Longrightarrow \quad u \sim_L v.$$

Proof. Because $L = L(M)$, we have

$$\begin{aligned} u^{-1}L &= \{w \mid uw \in L\} \\ &= \{w \mid \delta(q_0, uw) \in F\} \\ &= \{w \mid \delta(\delta(q_0, u), w) \in F\}. \end{aligned}$$

Clearly, the claim follows from the definitions of the two equivalence relations. □

For each language $L \subseteq \Sigma^*$ such that $\sim_L$ is of finite index, we define a DFA $M_L = (Q_L, \Sigma, \delta_L, i_L, F_L)$ as follows:

$$\begin{aligned} Q_L &= \{u^{-1}L \mid u \in \Sigma^*\}, \\ i_L &= \varepsilon^{-1}L = L, \\ F_L &= \{u^{-1}L \mid u \in L\} \text{ and} \\ \delta_L(u^{-1}L, a) &= a^{-1}(u^{-1}L) = (ua)^{-1}L. \end{aligned}$$

This automaton has an important role in the sequel. We are now ready for the Myhill-Nerode Theorem.

Theorem 12. *The following statements are equivalent.*

- *(a) L is a regular language;*
- *(b) There exists a right-invariant equivalence relation $\sim$ of finite index such that L is the union of some of its equivalence classes;*
- *(c) The Myhill-Nerode equivalence relation $\sim_L$ is of finite index and L is the union of some of its equivalence classes.*

Proof. First, suppose that L is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Now the equivalence relation $\sim_M$ satisfies the statement (b). It is seen to be right-invariant and the number of equivalence classes is clearly bounded by the number $|Q|$ of states. Also, $L = \bigcup_{\delta(q_0,w)\in F}[w]$.

Next, suppose that an equivalence relation $\sim$ satisfies the conditions (b). In order to prove that the Myhill-Nerode equivalence relations is of finite index it suffices to show that $\sim$ is a refinement of $\sim_L$. Let $u \sim v$ and $x \in u^{-1}L$. Then, by definition, $ux \in L$. Since L is a union of some equivalence classes of $\sim$, we must have $[ux] \subseteq L$. Now vx belongs to $[ux] \subseteq L$ by the right-invariance of the equivalence relation. Thus $x \in v^{-1}L$ and we have proved that $u^{-1}L \subseteq v^{-1}L$. By symmetry, $u^{-1}L = v^{-1}L$, which means that $u \sim_L v$. Also, L is a union of some of the equivalence classes of $\sim_L$. Namely, if $u \in L$ and $u \sim_L v$, then $\varepsilon \in u^{-1}L = v^{-1}L$, which means that $v \in L$. Thus $[u] \subseteq L$.

Finally, suppose that $\sim_L$ is of finite index. We prove that L is a regular language. Consider the automaton M_L defined above. This DFA accepts L, since

$$\begin{aligned} w \in L(M_L) &\iff \delta_L(L, w) \in F_L \\ &\iff w^{-1}L \in \{u^{-1}L \mid u \in L\} \\ &\iff \exists u \in L : w^{-1}L = u^{-1}L \\ &\iff w \in L. \end{aligned}$$

The last equivalence is based on the fact that $u \in L$ if and only if $\varepsilon \in u^{-1}L$. □

The connection between the previous theorem and minimization is established in the following theorem.

Theorem 13. *M_L is the unique (up to renaming of states) DFA accepting L with minimal number of states.*

Proof. Let L be accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$. We may suppose that the automaton M is *connected*, i.e. each state is *reachable* from the initial state by some word w. Removing from M those states which are not reachable from the initial state does not affect to the language accepted by the DFA.

Define a mapping $\nu \colon Q \to Q_L$ by setting:

$$\nu(q) = u^{-1}L, \text{ where } q = \delta(q_0, u).$$

We prove that the mapping is well defined. Let $\delta(q_0, u) = q = \delta(q_0, v)$ for $u, v \in \Sigma^*$. It follows from Lemma 2 that $u^{-1}L = v^{-1}L$. The definition of ν is therefore independent of the choice of the word u.

Clearly, ν is surjective. This means that any DFA accepting L must have at least $|Q_L|$ states. Thus M_L is minimal.

Assume now that also M is minimal. Then it has the same number of states as M_L and therefore ν is one-to-one. The initial state of M is mapped to the initial state of M_L:

$$\nu(q_0) = \nu(\delta(q_0, \varepsilon)) = \varepsilon^{-1}L = L = i_L. \tag{3}$$

Moreover, $q \in F$ if and only if $\nu(q) \in F_L$. Namely,

$$\begin{aligned} q \in F &\iff \exists u \in L : \delta(q_0, u) = q \\ &\iff \exists u \in L : \nu(q) = u^{-1}L \\ &\iff \nu(q) \in F_L. \end{aligned} \tag{4}$$

The first implication is based on the fact that the minimal automaton M is connected. Now the uniqueness of the minimal automaton follows from the following diagram expressing the renaming of states.

$$\begin{array}{ccc} q & \xrightarrow{a} & q' \\ {\scriptstyle\nu}\downarrow & & \downarrow{\scriptstyle\nu} \\ u^{-1}L & \xrightarrow{a} & (ua)^{-1}L \end{array}$$

Let $\delta(q_0, u) = q$ and $\delta(q, a) = q'$. Then

$$\delta_L(\nu(q), a) = \delta_L(u^{-1}L, a) = (ua)^{-1}L = \nu(q') = \nu(\delta(q, a)) \tag{5}$$

proving the diagram. □

There are many practical algorithms for the minimization of deterministic automata. For example, Moore's and Hopcroft's algorithms are based on consecutive refinements of partitions of the set of states by means of Myhill-Nerode equivalence relation [17, 14]. Brzozowski's algorithm minimizes an automaton by performing reversal and determinization twice [6]. Intuitively basic algorithm due to Moore computes the minimal automaton in time $\mathcal{O}(n^3k)$, where n is the number of states of the given automaton and k is the size of the input alphabet. The best known complexity $\mathcal{O}(n \log n)$ is obtained by Hopcroft's algorithm. Brzozowski's algorithm has exponential complexity, since determinization may produce exponential number of states, but in practice the algorithm seems to have exceptionally good performance; see [32]. The implementation of Hopcroft's and Brzozowski's algorithms can be found, for example, in Grail+, which is a symbolic computation environment; see `http://www.csd.uwo.ca/research/grail/`.

Finally, we show how minimization can be applied to finite automata with output. A DFAO with a finite-state function f is called *minimal* if it has the smallest possible number of states among all DFAOs generating this finite-state function. Suppose that a finite-state function f is defined by an automaton $M_f = (Q, \Sigma, \delta, q_0, \Delta, \tau)$. Our aim is to modify the DFAO M_f to a DFA M, minimize M to M_L and modify this minimal DFA to a minimal DFAO $\overline{M}_f$:

$$\text{DFAO } M_f \longmapsto \text{DFA } M \longmapsto \text{DFA } M_L \longmapsto \text{DFAO } \overline{M}_f$$

First we show how these automata are constructed. Let $|\Delta| \geq 2$. Otherwise the function and the automaton are trivial. Assume also that $\Sigma \cap \Delta = \emptyset$ by renaming the symbols if needed. Furthermore, let M_f be connected. We now define a DFA

$$M = (Q \cup \{f, g\}, \Sigma \cup \Delta, \delta', q_0, \{f\}),$$

where f is a new special accepting state and g is a new garbage state. The transi-

tion function δ' is defined by

$$\begin{aligned}
\delta'(g,a) &= g,\\
\delta'(f,a) &= g,\\
\delta'(q,a) &= \begin{cases} \delta(q,a) & \text{if } a \in \Sigma,\\ f & \text{if } a \in \Delta \text{ and } \tau(q) = a,\\ g & \text{if } a \in \Delta \text{ and } \tau(q) \neq a \end{cases}
\end{aligned}$$

for all $a \in \Sigma \cup \Delta$ and for $q \in Q$. Note that also M is connected. Let $L = L(M)$. Clearly, the language L consists of words of the form wd, where $w \in I_d(M)$ and $d \in \Delta$. In other words,

$$L = \bigcup_{d \in \Delta} I_d(M)\{d\}.$$

The minimal automaton accepting the language L is denoted by

$$M_L = (Q_L,\, \Sigma \cup \Delta,\, \delta_L,\, i_L,\, F_L).$$

Note that $Q_L = \nu(Q \cup \{f, g\})$. Also $F_L = \{\nu(f)\}$ by the equalities in (4). We convert this DFA M_L to a DFAO

$$\overline{M}_f = (\overline{Q},\, \Sigma,\, \overline{\delta},\, i_L,\, \Delta,\, \overline{\tau})$$

as follows. The set of states $\overline{Q}$ is $\nu(Q)$. For every state $\overline{q} \in \overline{Q}$ and every letter $a \in \Sigma$, we define $\overline{\delta}(\overline{q}, a) = \delta_L(\overline{q}, a)$. Finally we define that $\tau(\overline{q}) = a$, where $a \in \Delta$ is the unique symbol satisfying $\delta_L(\overline{q}, a) = \nu(f)$.

Next we prove that $\overline{M}_f$ is well-defined. This concerns the definitions of functions $\overline{\delta}$ and $\overline{\tau}$. Since, for every $a \in \Sigma$ and $q \in Q$, $\delta'(q, a)$ belongs to Q, we have, by (5),

$$\delta_L(\nu(q), a) = \nu(\delta'(q, a)) \in \nu(Q).$$

Thus $\overline{\delta}$ is a mapping from $\overline{Q} \times \Sigma$ to $\overline{Q}$. In addition, we have to show that for every $\overline{q} \in \nu(Q)$ there always exists a unique symbol $a \in \Delta$ such that $\delta_L(\overline{q}, a) = \nu(f)$. Suppose that $\overline{q} = \nu(q)$ for some $q \in Q$ and $\delta_L(\overline{q}, a) = \delta_L(\overline{q}, b) = \nu(f)$, where $a, b \in \Delta$ and $a \neq b$. Now using (5) we have

$$\delta_L(\nu(q), a) = \nu(f) \iff \nu(\delta'(q, a)) = \nu(f) \iff \delta'(q, a)) = f,$$

where the last equivalence is based on the fact that $\nu^{-1}(\nu(f)) = \{f\}$ following from (4). Thus $\delta'(q, a) = f$ and $\delta'(q, b) = f$. By the structure of M, this is a contradiction and hence τ is well-defined. We may now state the following theorem.

Theorem 14. *$\overline{M}_f$ is the unique (up to renaming of states) minimal DFAO with finite-state function f.*

Proof. First we have to prove that $\overline{M}_f$ is an automaton with finite-state function f. Suppose that $f(w) = a$ for a word $w \in \Sigma^*$ and letter $a \in \Delta$. This means that wa belongs to the language L. By the definition of M, $\delta(q, a) = f$ for a state $q = \delta(q_0, w)$. Using (3) and (5) it follows that $\delta_L(i_L, w) = \nu(q) \in \nu(Q)$ and $\delta_L(\nu(q), a) = \nu(f)$. Thus $\overline{\tau}(\overline{\delta}(i_L, w)) = \overline{\tau}(\nu(q)) = a$.

Let us then consider the number of states in $\overline{M}_f$. Since $\nu(q)$ is the language consisting of those words w such that $\delta'(q, w) \in F$, we have $\nu(f) = \{\varepsilon\}$, $\nu(g) = \emptyset$ and, for every $q \in Q$, there exists a letter $a \in \Delta$ such that $a \in \nu(q)$. Thus $\overline{Q} = \nu(Q) = Q_L \setminus \{\nu(f), \nu(q)\}$ and $|\overline{Q}| = |Q_L| - 2$. Suppose now that $|Q_L| = n$. If $\overline{M}_f$ were not minimal, then there would exist an automaton M'_f with less than $n - 2$ states. Using the procedure above we would obtain a DFA M' accepting the language L with less than n states. This contradicts with the minimality of M_L. The uniqueness of $\overline{M}_f$ can be seen similarly as in Theorem 13. Namely, a state $q \in Q$ is mapped to a state $\nu(q) \in \overline{Q}$ and a transition $\delta(q, a)$, where $a \in \Sigma$, corresponds to a transition $\overline{\delta}(\nu(q), a)$. □

2 Automatic sequences

In this section we define automatic sequences, consider a couple of examples and prove some basic properties of these sequences. In Sections 2.2–2.4 we give three different characterizations of automatic sequences, which connect them to uniform morphisms and regular languages. In the last two subsections we prove some important closure properties and consider the subword complexity of automatic sequences.

2.1 Definition and examples

Let $k \geq 2$ be an integer. Recall that a deterministic automaton with output is a k-DFAO if the input alphabet is $\Sigma_k = \{0, 1, \ldots, k-1\}$. Recall also the notations $(n)_k$ and $[w]_k$ defined by the equations (1) and (2).

Definition 1. A sequence $(a_n)_{n\geq 0}$ over a finite alphabet Δ is *k-automatic* if there exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, w))$ for all $n \geq 0$ and for all $w \in \Sigma_k^*$ with $[w]_k = n$.

The automaton M is said to *generate* the sequence $(a_n)_{n\geq 0}$. This means that any digit a_n can be computed using M. We feed the k-DFAO with the base-k representation of an integer n starting from the most significant digit and get a_n as an output. Note that the automaton functions properly also in the case when there are extra zeros in the beginning of the representation. In this sense Definition 1 is robust. The set of automatic sequences could equivalently be defined so that only normalized representations were considered. Suppose that $(a_n)_{n\geq 0}$ is generated by a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, (n)_k))$ for all $n \geq 0$. The sequence is seen to be automatic by slightly modifying M. We add a new state q_0' such that $\delta(q_0', 0) = q_0'$ and $\delta(q_0', i) = \delta(q_0, i)$ for $i \neq 0$. The output is defined by $\tau(q_0') = \tau(q_0)$. Now $\tau(\delta(q_0', 0^i(n)_k)) = \tau(\delta(q_0, (n)_k))$ for all $i \geq 0$. Hereby, we may always assume that $\delta(q_0, 0) = 0$. Another relaxation in the definition of automatic sequences concerns the way of processing input digits. Namely, the input can be read starting from the least significant digit as well. As a consequence of Theorem 11 we obtain:

Theorem 15. *The sequence $(a_n)_{n\geq 0}$ is k-automatic if and only if there exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, w^R))$ for all $n \geq 0$ and all $w \in \Sigma_k^*$ with $[w]_k = n$.*

Many sequences of mathematical interest can be shown to be k-automatic. We give a few famous examples.

Example 2. (The Thue-Morse sequence) This sequence $\mathbf{t} = (t_n)_{n\geq 0}$ counts the number of 1's (mod 2) in the base-2 representation of n. The first few terms of the Thue-Morse sequence are

$$\mathbf{t} = 0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,1\,0\,0\,1\,0\cdots$$

For example, $(5)_2 = 101$ and $|101|_1 = 2$. Thus $|(5)_2|_1 \equiv 0 \pmod 2$ and $t_5 = 0$. It is easy to see that the Thue-Morse sequence is generated by the DFAO in Figure 4.

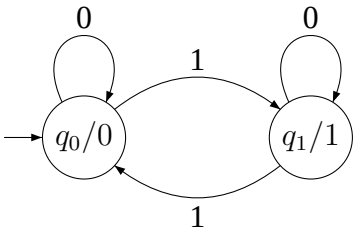


Figure 4: Automaton generating the Thue-Morse sequence

The state q_0 corresponds to an even number of 1's and q_1 to an odd number of 1's. If 1 occurs in the input, the parity of the number of 1's changes and the transition is from the current state to the other. If the input is 0, then the parity remains unchanged and the transition is from the state into itself. Note that $\mathbf{t}$ is a special case of sequences $(s_k(n) \pmod m)_{n\geq 0}$, where $(n)_k = b_0 b_1 \cdots b_r$ and $s_k(n) = \sum_{i=0}^{r} b_i$. Clearly, $\mathbf{t} = (s_2(n) \pmod 2)$. In general, the sequence $(s_k(n) \pmod m)_{n\geq 0}$ is k-automatic for $k \geq 2,\ m \geq 1$. The generating automaton is $A_m = (Q_m,\ \Sigma_k,\ \delta_m,\ q_0, \Sigma_m, \tau_m)$, where

$$\begin{aligned} Q_m &= \{q_i \mid i = 0, 1, \ldots, m-1\}, \\ \delta(q_i, a) &= q_j, \text{where } j = i + a \pmod m, \\ \tau(q_i) &= i. \end{aligned}$$

The Thue-Morse sequence occurs in many areas of mathematics: combinatorics on words, differential geometry, number theory and mathematical physics; see [3]. We mention here as an example the *multigrades* problem of Prouhet [24]: Given positive integer t, is it possible to partition the set $\{0, 1, 2, \cdots, 2^N - 1\}$ into two disjoint sets A and B such that

$$\sum_{a\in A} a^i = \sum_{b\in B} b^i$$

for all $i = 0, 1, \ldots, t$? By convention, $0^0 = 1$. The Thue-Morse sequence gives an answer for the case $N = t + 1$. Namely, choose

$$\begin{aligned} A &= \{0 \leq j < 2^{t+1} \mid t_j = 0\}, \\ B &= \{0 \leq j < 2^{t+1} \mid t_j = 1\}. \end{aligned}$$

For example, $0^i + 3^i + 5^i + 6^i = 1^i + 2^i + 4^i + 7^i$ for $i = 0, 1, 2$.

Example 3. (The Rudin-Shapiro sequence) Another famous 2-automatic sequence is the Rudin-Shapiro sequence $\mathbf{r} = (r_n)_{n\geq 0}$, where $r_n = (-1)^{e_{2,11}(n)}$ and $e_{2,11}(n)$ is the number of (possibly overlapping) occurrences of 11 in the canonical base-2 representation of n. The sequence begins by

$$\mathbf{r} = 1\,1\,1\,\text{-}1\,1\,1\,\text{-}1\,1\,1\,1\,1\,\text{-}1\cdots .$$

The 2-DFAO generating the sequence is represented in Figure 5. The states are labeled by words ab, where a denotes the parity of the number of occurrences of the pattern 11 seen so far and b is the last digit symbol. The output of a state ab is $(-1)^a$. The transitions are easily obtained with the help of this interpretation. This sequence is related, for example, to harmonic analysis; see [31].

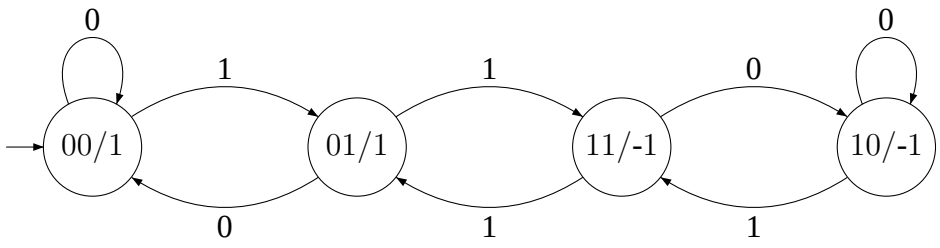


Figure 5: Automaton generating the Rudin-Shapiro sequence

Not all automatic sequences with mathematical interest are 2-automatic. As an example, consider the following 3-automatic sequence.

Example 4. (The Schröder numbers) Let S_n be the nth Schröder number, i.e. the number of paths in $\mathbb{Z}\times\mathbb{Z}$ from $(0,0)$ to (n,n) composed of steps $\{(0,1),(1,0),(1,1)\}$ and containing no points above the line $x = y$.
The sequence $\mathbf{s} = (S_n \bmod 3)_{n\geq 0}$ is 3-automatic. The automaton generating it is represented in Figure 7. The few first terms of this sequence are

$$\mathbf{s} = 1\,2\,0\,1\,0\,1\,0\,2\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,2\,0\cdots .$$

See the solutions of a problem in [33] for detailed analysis of this sequence.

For other examples of automatic sequences see the survey by J.-P. Allouche [2]. We notice that there exist also sequences, which are not automatic in any base $k \geq 2$. For example, the characteristic sequence of squares and the characteristic sequence of primes are such sequences. For more details about these nonautomatic sequences; see [5, 20, 27].

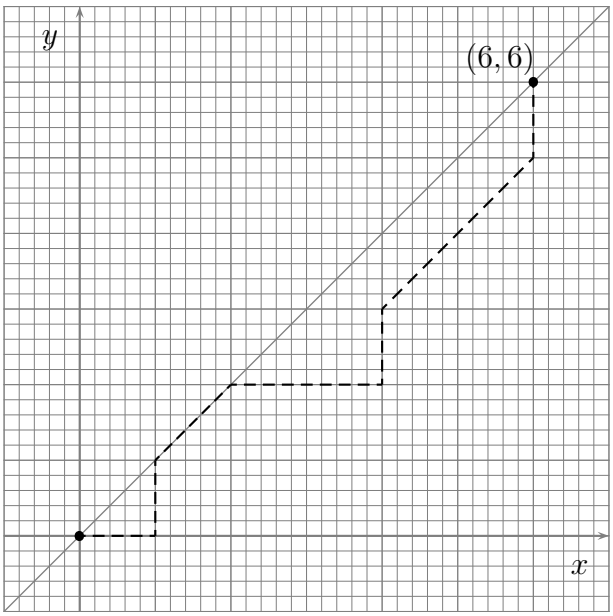


Figure 6: An example of a Schröder path from (0,0) to (6,6).

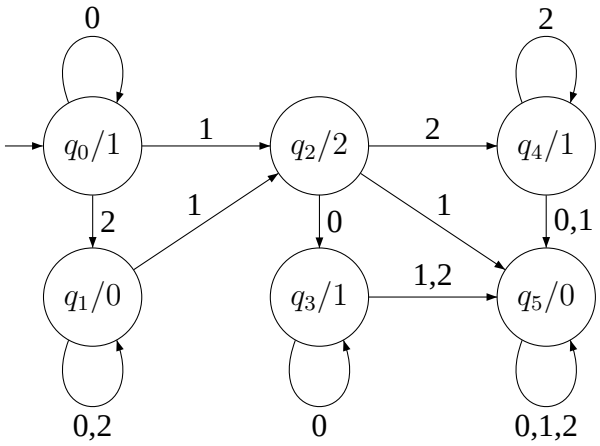


Figure 7: Automaton generating the Schröder numbers modulo 3

2.2 Fibers

In this subsection we give the first alternative characterization of automatic sequences. It establishes a relation between automatic sequences and regular languages. Recall that a language is regular if and only if it is rational by Kleene's result (Theorem 2). Let $\mathbf{a} = (a_n)_{n\geq 0}$ be a sequence over Δ, let $k \geq 2$ be an integer and $d \in \Delta$. We consider the sets $I_k(\mathbf{a}, d) = \{(n)_k \mid a_n = d\}$, which are called *$k$-fibers*.

Theorem 16. *Sequence* $\mathbf{a} = (a_n)_{n\geq 0}$ *is* k*-automatic if and only if each of the fibers* $I_k(\mathbf{a}, d)$ *is a regular language for all* $d \in \Delta$*.*

Proof. Let $(a_n)_{n\geq 0}$ be generated by the k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$. By Theorem 9, the language $I_d(M) = \{w \in \Sigma_k^* \mid \tau(\delta(q_0, w)) = d\}$ is regular. Recall that $C_k = \{\varepsilon\} \cup (\Sigma_k \backslash \{0\})\Sigma^*$. Now $I_k(\mathbf{a}, d) = I_d(M) \cap C_k$ and it is regular by Theorem 3.

Conversely, suppose that $I_k(\mathbf{a}, d)$ is regular for all $d \in \Delta$. Then also $I'_k(d) = 0^* I_k(\mathbf{a}, d)$ is regular. Let $d, d' \in \Delta$ and $d \neq d'$. If $w \in I'_k(d) \cap I'_k(d')$, then $d = \tau(\delta([w]_k)) = d'$. This is impossible. Thus the sets $I'_k(d)$ are disjoint and they partition Σ^*. We conclude that $(a_n)_{n\geq 0}$ is k-automatic using Theorem 10. □

We give a short example.

Example 5. Consider the fiber $I_2(\mathbf{t}, 1)$ of the Thue-Morse sequence $\mathbf{t}$. It is the language consisting of the words in the binary alphabet $\{0, 1\}$ with an odd number of ones. It is regular by the previous theorem. Actually, $I_2(\mathbf{t}, 1) = 0^*1(10^*1 + 0^*)^* = 0^*10^*(10^*1)^*0^*$.

Let $L \subseteq \Sigma_k^*$ be a regular language. We modify it by the representation of a single number n, more precisely by the language $0^*(n)_k$. By Theorem 3, also the languages $L \cup 0^*(n)_k$ and $L \setminus 0^*(n)_k$ are regular. Thus applying a finite number of such modifications by languages $0^*(n)_k$ to fibers $I_k(\mathbf{a}, d)$ we obtain the following corollary.

Corollary 2. *If a sequence* $(b_n)_{n\geq 0}$ *differs only in finitely many digits from a* k*-automatic sequence* $(a_n)_{n\geq 0}$*, then it is* k*-automatic.*

2.3 Kernels

The next characterization is due to Eilenberg [10, Prop. V.3.3]. The proof is represented in [4, Theorem 6.6.2].

Definition 2. Let $\mathbf{u}$ be an infinite sequence. The *k-kernel* of $\mathbf{u}$ is the set of subsequences

$$K_k(\mathbf{u}) = \{(\mathbf{u}(k^i n + j))_{n\geq 0} \mid i \geq 0, 0 \leq j < k^i\}.$$

Theorem 17. *Let* $k \geq 2$*. The sequence* $\mathbf{u}$ *is* k*-automatic if and only if* $K_k(\mathbf{u})$ *is finite.*

Proof. Suppose that $\mathbf{u}$ is k-automatic. By Theorem 15 there exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $\mathbf{u}(n) = \tau(\delta(q_0, (n)_k^R 0^t))$ for all $t \geq 0$. Thus the output does not depend on the number of zeros in the end of the input. Let

$q = \delta(q_0, w^R)$, where $w = 0^t(j)_k$ for some nonnegative integers t and j. Denote $|w| = i$. For $n > 0$ we have

$$\begin{aligned}\delta(q_0, (k^i \cdot n + j)_k^R) &= \delta(q_0, ((n)_k w)^R) \\ &= \delta(\delta(q_0, w^R), (n)_k^R) \\ &= \delta(q, (n)_k^R).\end{aligned}$$

Secondly, suppose that $n = 0$. Then

$$\begin{aligned}\delta(q_0, (k^i \cdot n + j)_k^R) &= \delta(q_0, (j)_k^R) = \delta(q_0, (j)_k^R 0^t) \\ &= \delta(q_0, w^R) = q \\ &= \delta(q, \varepsilon) = \delta(q, (0)_k^R).\end{aligned}$$

Hence $\mathbf{u}(k^i \cdot n + j) = \tau(\delta(q_0, (k^i \cdot n + j)_k^R) = \tau(\delta(q, (n)_k^R)$ for all $n \geq 0$. Since there are only finitely many choices of q, the k-kernel $K_k(\mathbf{u})$ must be finite.

Conversely, suppose that $K_k(\mathbf{u})$ is finite. Define a relation

$$w \sim v \iff \mathbf{u}(k^{|w|} \cdot n + [w]_k) = \mathbf{u}(k^{|v|} \cdot n + [v]_k)$$

for all $n \geq 0$. This relation is clearly an equivalence relation. By the assumption, it partitions Σ^* into a finite number of equivalence classes $[w]$. Define a k-DFAO as follows:

$$\begin{aligned}Q &= \{[w] \mid w \in \Sigma_k^*\}, \\ \delta([w], a) &= [aw], \\ \tau([w]) &= \mathbf{u}([w]_k), \\ q_0 &= [\varepsilon].\end{aligned}$$

We prove that this automaton is well-defined, i.e. independent of the choice of representatives. Assume that $[w] = [v]$. We have to show that

$$\begin{aligned}&(1) \quad \delta([w], a) = \delta([v], a) \quad \text{for each } a \in \Sigma_k \text{ and} \\ &(2) \quad \tau([w]) = \tau([v]).\end{aligned}$$

Substituting $n = km + a$ into the definition of the equivalence relation we obtain

$$\mathbf{u}(k^{|aw|} \cdot m + [aw]_k) = \mathbf{u}(k^{|av|} \cdot m + [av]_k)$$

for all $m \geq 0$. Therefore (1) is true, and (2) follows by substituting $n = 0$. Next we prove by induction on the length of w that $\delta(q_0, w^R) = [w]$. The case $w = \varepsilon$ is trivial. Assume now that the claim holds for all the words of length n. Suppose that $w = av$, where $|v| = n$. Then $\delta(q_0, w^R) = \delta(\delta(q_0, v^R), a) = \delta([v], a) = [av]$. By the definition of τ we therefore have $\tau(\delta(q_0, w^R)) = \mathbf{u}([w]_k)$. It follows from Theorem 15 that $\mathbf{u}$ is a k-automatic sequence. □

Example 6. We count the 2-kernel of the Rudin-Shapiro sequence $\mathbf{r} = (r_n)_{n\geq 0}$ of Example 3. Consider the subsequences $(r_{2^i n+j})_{n\geq 0}$. We can find equal subsequences by considering the number of occurrences of 11 in the presentations of $(n)_k$ and $(2^i n + j)_k$. For example, $(4n)_2 = (n)_2 00$ and therefore $r_{4n} = r_n$. Similarly, since $(2n + 1)_2 = (n)_2 1$ and $(8n + 5)_2 = (n)_2 101$, the number of factors 11 is equal and thus $r_{2n+1} = r_{8n+5}$. In this way we finally obtain the kernel $K_2(\mathbf{r}) = \{(r_n)_{n\geq 0}, (r_{2n+1})_{n\geq 0}, (r_{4n+3})_{n\geq 0}, (r_{8n+3})_{n\geq 0}\}$.

2.4 Uniform morphisms

Automatic sequences are closely related to fixed points of uniform morphisms. First we recall some basic concepts. Let $\varphi\colon \Sigma^* \to \Sigma^*$ be a morphism on Σ^*. A finite or infinite word w satisfying $\varphi(w) = w$ is said to be a *fixed point of* φ. If there exists a letter $a \in \Sigma$ such that $\varphi(a) = ax$ and $\varphi^k(x) \neq \varepsilon$ for all $k \geq 0$, we say that φ is *prolongable* on a. In this case, we can define an infinite sequence $\varphi^\omega(a) := ax\varphi(x)\varphi^2(x)\cdots$, which clearly is a fixed point of φ. If $\mathbf{w} = \varphi^\omega(a)$, then we call $\mathbf{w}$ a *pure morphic sequence* and say that $\mathbf{w}$ is *generated* by φ. If there is a coding $\tau\colon \Sigma^* \to \Delta^*$ and $\mathbf{w} = \tau(\varphi^\omega(a))$, then we call $\mathbf{w}$ a *morphic sequence* generated by φ and τ. If the morphism φ is uniform, then we talk about *uniformly morphic sequences*.

A useful characterization of automatic sequences in terms of k-uniform morphisms and codings is known as (the second) Cobham's theorem. Another famous theorem by Cobham is presented in Section 3. We begin by a technical lemma.

Lemma 3. *Suppose that* $\mathbf{w} = \varphi(\mathbf{w}) = a_0 a_1 a_2 \cdots$ *for some* k*-uniform morphism* φ*. Then* $\varphi(a_i) = a_{ki} a_{ki+1} \cdots a_{ki+k-1}$.

Proof. Since $\varphi(\mathbf{w}) = \mathbf{w}$ and φ is k-uniform, then

$$\varphi(a_0 a_1 \cdots a_i) = a_0 a_1 \cdots a_{ki+k-1}.$$

Hence $\varphi(a_i)$ must be the last k letters of $\varphi(a_0 a_1 \cdots a_i)$, i.e. $\varphi(a_i)(j) = a_{ki+j}$ for all $0 \leq j \leq k-1$. □

Theorem 18. (Cobham) *Let* $k \geq 2$*. Then a sequence* $\mathbf{u} = (u_n)_{n\geq 0}$ *is* k*-automatic if and only if it is the image, under a coding, of a fixed point of a* k*-uniform morphism.*

Proof. Suppose that $\mathbf{u} = \tau(\mathbf{w})$, where $\tau\colon \Sigma^* \to \Delta^*$ is a coding, and $\mathbf{w} = \varphi(\mathbf{w}) = w_0 w_1 w_2 \cdots$, where $\varphi\colon \Sigma^* \to \Sigma^*$ is a k-uniform morphism. Define a k-DFAO $M = (\Sigma, \Sigma_k, \delta, w_0, \Delta, \tau)$, where $\delta(q, b) = \varphi(q)(b)$, i.e. the $(b+1)$th letter of the word $\varphi(q)$. We claim that $w_n = \delta(w_0, (n)_k)$ for all $n \geq 0$. This is proved by induction on n. The case $n = 0$ is clear. Suppose that the claim holds for all nonnegative integers $i < n$. We prove it for n. Let $(n)_k = n_0 n_1 \cdots n_t$,

where $n_0 n_1 \cdots n_{t-1} = (n')_k$ and $n = n'k + n_t$. Then

$$\begin{aligned}
\delta(w_0, (n)_k) &= \delta(w_0, n_0 n_1 \cdots n_t) \\
&= \delta(\delta(w_0, n_0 n_1 \cdots n_{t-1}), n_t) \\
&= \delta(\delta(w_0, (n')_k), n_t) \\
&= \delta(w_{n'}, n_t) \quad \text{(by the induction hypothesis)} \\
&= \varphi(w_{n'})(n_t) \\
&= w_{kn'+n_t} \quad \text{(by Lemma 3)} \\
&= w_n.
\end{aligned}$$

Thus $u_n = \tau(w_n) = \tau(\delta(w_0, (n)_k))$, and therefore $\mathbf{u}$ is k-automatic.

Conversely, suppose that $\mathbf{u}$ is generated by a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$. As was noted in the beginning of Section 2.1 we may assume without loss of generality that $\delta(q_0, 0) = q_0$. Now define the morphism φ as follows:

$$\varphi(q) = \delta(q, 0)\delta(q, 1) \cdots \delta(q, k-1),$$

for each $q \in Q$. Since $\delta(q_0, 0) = q_0$, the morphism φ is prolongable on q_0, and hence the fixed point $\varphi^\omega(q_0) = \mathbf{w} = w_0 w_1 w_2 \cdots$ exists. We prove by induction on $|x|$ that $\delta(q_0, x) = w_{[x]_k}$ for all $x \in \Sigma_k^*$. If $|x| = 0$, then $\delta(q_0, \varepsilon) = q_0 = w_0 = w_{[\varepsilon]_k}$. Secondly, suppose that the claim holds for all words $y \in \Sigma_k^*$, where $|y| < n$. Let $x = ya$ be a word, where y belongs to Σ_k^{n-1} and $a \in \Sigma_k$. Then

$$\begin{aligned}
\delta(q_0, x) &= \delta(q_0, ya) \\
&= \delta(\delta(q_0, y), a) \\
&= \delta(w_{[y]_k}, a) \quad \text{(by the induction hypothesis)} \\
&= \varphi(w_{[y]_k})(a) \quad \text{(by the definition of } \varphi) \\
&= w_{k \cdot [y]_k + a} \quad \text{(by Lemma 3)} \\
&= w_{[ya]_k} \\
&= w_{[x]_k}.
\end{aligned}$$

Since $[(n)_k]_k = n$, we have $u_n = \tau(\delta(q_0, (n)_k)) = \tau(w_n)$ and $\mathbf{u} = \tau(\varphi^\omega(q_0))$. □

Using the construction of the previous theorem we have the following example.

Example 7. We represent the sequence $(S_n \bmod 3)_{n \geq 0}$, where S_n is the nth Schröder number, as a fixed point of a 3-uniform morphism φ under a coding τ. From the automaton of Example 4, see page 21, we have in a new alphabet $\{A, B, \ldots, F\}$

$$\begin{array}{lcl@{\qquad}lcl}
\varphi: & A & \mapsto ABC & \tau: & A & \mapsto 1 \\
& B & \mapsto DEF & & B & \mapsto 2 \\
& C & \mapsto CBC & & C & \mapsto 0 \\
& D & \mapsto DEE & & D & \mapsto 1 \\
& E & \mapsto EEE & & E & \mapsto 0 \\
& F & \mapsto EEF & & F & \mapsto 1
\end{array}$$

and the fixed point is

$$\tau(\varphi^\omega(A)) = \tau(A\,B\,C\,D\,E\,F\,C\,B\,C\,D\,E\,E\cdots) = 1\,2\,0\,1\,0\,1\,0\,2\,0\,1\,0\,0\cdots,$$

which is exactly the sequence $\mathbf{s}$.

By applying the morphic characterization of Theorem 18 and the k-fiber characterization of Section 2.2 we have the following theorem.

Theorem 19. *For all $m \geq 1$, a sequence $\mathbf{u} = (u_n)_{n\geq 0}$ is k-automatic if and only if it is k^m-automatic.*

Proof. Suppose first that $\mathbf{u}$ is k-automatic. By Theorem 18 we may write it in the form

$$\mathbf{u} = \tau(\varphi^\omega(a)),$$

where τ is a coding, φ is a k-uniform morphism and a is a letter. Now define $\phi = \varphi^m$. Clearly,

$$\mathbf{u} = \tau(\phi^\omega(a)).$$

Since ϕ is k^m-uniform, also $\mathbf{u}$ is k^m-automatic by Theorem 18.

Assume conversely that $\mathbf{u} = (u_n)_{n\geq 0}$ is k^m-automatic. By Theorem 16, the language

$$I_{k^m}(\mathbf{u}, d) = \{(n)_{k^m} \mid u_n = d\}$$

is regular for all $d \in \Delta$. Now define morphism $\beta\colon \Sigma_{k^m}^* \to \Sigma_k^*$ such that for each $0 \leq j \leq k^m - 1$, $\beta(j) = w$, where $|w| = m$ and $[w]_k = j$. By induction on the length of w we see that for any nonnegative integer n, $[\beta((n)_{k^m})]_k = n$. If $|(n)_{k^m}| = 1$, then $(n)_{k^m} = n$ and the claim follows from the definition of β. Suppose that $(n)_{k^m} = n_0 n_1 \cdots n_l$, where $n_0 \cdots n_{l-1} = (n')_{k^m}$, $n = n'k^m + n_l$ and $[\beta((n')_{k^m})]_k = n'$. Since the length of the image $\beta(j)$ is always m, we have $[\beta((n)_{k^m})]_k = [\beta((n')_{k^m})\beta(n_l)]_k = [\beta((n')_{k^m})]_k \cdot k^m + [\beta(n_l)]_k = n' \cdot k^m + n_l = n$. Recall that $C_k = \{\varepsilon\} \cup (\Sigma_k \setminus \{0\})\,\Sigma_k^*$. By Theorem 5, $\beta(I_{k^m}(\mathbf{u}, d))$ is regular for all d, and hence

$$(0^*)^{-1}\beta(I_{k^m}(\mathbf{u}, d)) \cap C_k = \{(n)_k \mid u_n = d\}$$

is also regular. By Theorem 16, $\mathbf{u}$ is k-automatic. □

Sequences which are both k- and l-automatic, for two bases k and l, are considered in more details with respect to subword complexity and periodicity in Section 2.6 and also in Section 3.

2.5 Closure properties

In the article [8] Cobham mentions several closure properties of automatic sequences. The class of k-automatic sequences is closed, for example, under shift, periodic deletion, q-block substitution and q-block deletion. We follow here the

formulation of closure properties by Allouche and Shallit [4]. In the end of this subsection we show how these properties can be easily used to prove Cobham's closure results.

The next two theorems concern subsequences of automatic sequences. They are kind of converses to each other.

Theorem 20. *Let $\mathbf{u} = (u_n)_{n\geq 0}$ be a k-automatic sequence. Then for all integers $a,\ b \geq 0$ the subsequence $(u_{an+b})_{n\geq 0}$ is also k-automatic.*

Proof. The case $a = 0$ is trivial, since then $(u_{an+b})_{n\geq 0} = (u_b)_{n\geq 0}$ is a sequence in one letter alphabet and clearly automatic. Let $a \geq 1$. We use the characterization of Theorem 17. Suppose that the finite k-kernel of $\mathbf{u}$ is

$$K_k(\mathbf{u}) = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$$

for some $r \geq 1$. Next we define a finite set

$$S = \{(\mathbf{u}_i(an + c))_{n\geq 0} \mid 1 \leq i \leq r, 0 \leq c < a + b\}.$$

Our aim is to prove that the k-kernel of $\mathbf{v} = (\mathbf{u}(an + b))_{n\geq 0}$ is a subset of S. Consider a sequence $(\mathbf{v}(k^e n + j))_{n\geq 0}$, where $0 \leq j < k^e$ and $e \geq 0$. Using the division algorithm we can find nonnegative integers d and f such that

$$ja + b = dk^e + f, \tag{6}$$

where $0 \leq f < k^e$. In addition, the inequality $ja + b < k^e a + b \leq k^e(a + b)$ implies that $0 \leq d < a + b$. Now, for all $n \geq 0$, using equation (6) we have

$$\mathbf{v}(k^e n + j) = \mathbf{u}(a(k^e n + j) + b) = \mathbf{u}(k^e an + ja + b) = \mathbf{u}(k^e(an + d) + f). \tag{7}$$

By the definition of the k-kernel,

$$(\mathbf{u}(k^e m + f))_{m\geq 0} = (\mathbf{u}_i(m))_{m\geq 0} \tag{8}$$

for some i, $1 \leq i \leq r$. Hence combining (7) and (8) with $m = an + d$ we get

$$(\mathbf{v}(k^e n + j))_{n\geq 0} = (\mathbf{u}(k^e(an + d) + f))_{n\geq 0} = (\mathbf{u}_i(an + d))_{n\geq 0} \in S.$$

Hence $K_k(\mathbf{v}) \subset S$ is finite and, by Theorem 2.2, $\mathbf{v}$ is automatic. □

Theorem 21. *Let a be a positive integer, and let $(u_n)_{n\geq 0}$ be a sequence such that $(u_{an+i})_{n\geq 0}$ is k-automatic for $0 \leq i < a$. Then $(u_n)_{n\geq 0}$ is itself k-automatic.*

Proof. By Theorem 16, the fiber

$$\{(n)_k \mid u_{an+i} = d\}$$

is a regular language for each $d \in \Delta$ and $0 \leq i < a$. Using Theorem 8 and Corollary 1 we see that also the language

$$X_{i,d} = \{(an + i)_k \mid u_{an+i} = d\}$$

is regular. Since finite unions of regular languages are regular (Theorem 3),

$$Y_d = \bigcup_{i=0}^{a-1} X_{i,d} = \{(n)_k \mid u_n = d\}$$

is regular for each $d \in \Delta$. Using again the k-fiber characterization (Theorem 16) we conclude that $(u_n)_{n\geq 0}$ is k-automatic. □

Let us proceed by considering automatic sequences under different mappings. We start with an easy theorem.

Theorem 22. *Let $\mathbf{u} = (u_n)_{n\geq 0}$ be a k-automatic sequence and let ρ be a coding. Then the sequence $\rho(\mathbf{u})$ is also k-automatic.*

Proof. Replace the output function τ of the automaton generating $\mathbf{u}$ by the function $\rho \circ \tau$. □

As a consequence of the previous theorems we have

Corollary 3. *Let $\mathbf{u} = (u_n)_{n\geq 0}$ be a k-automatic sequence, and let ψ be an l-uniform morphism for some $l \geq 1$. Then $\psi(\mathbf{u})$ is also k-automatic.*

Proof. Suppose that $\mathbf{u}$ is a k-automatic sequence over the alphabet Δ and $\psi\colon \Delta^* \to (\Delta')^*$ is an l-uniform morphism. We define a coding ψ_i by the rule

$$\psi_i(a) = \psi(a)(i).$$

for each $0 \leq i \leq l-1$. By the previous theorem, the sequences $\psi_i(\mathbf{u})$ are k-automatic. Clearly, for $0 \leq i < l$ and $n \geq 0$, $\psi(\mathbf{u})(ln+i) = \psi(u_n)(i) = \psi_i(u_n)$, and thus $(\psi(\mathbf{u})(ln+i))_{n\geq 0} = \psi_i(\mathbf{u})$. The result follows then from Theorem 21. □

Consider next cartesian products of automatic sequences.

Theorem 23. *Suppose that a sequence $\mathbf{a}_i$ over an alphabet Δ_i is k-automatic for each $i = 1, 2, \ldots, l$. Let f be any function from $\Delta_1 \times \Delta_2 \times \cdots \times \Delta_l$ to an alphabet Δ'. Then the sequence $(f(\mathbf{a}_1(n), \mathbf{a}_2(n), \ldots, \mathbf{a}_l(n)))_{n\geq 0}$ is k-automatic.*

Proof. Suppose that $\mathbf{a}_i$ is generated by a DFAO $M_i = (Q_i, \Sigma_k, \delta_i, q_{0i}, \Delta_i, \tau_i)$ for $1 \leq i \leq l$. Then $M' = (Q_1 \times Q_2 \times \cdots \times Q_l, \Sigma_k, \delta', (q_{01}, q_{02}, \ldots, q_{0l}), \Delta', \tau')$ generates $(f(\mathbf{a}_1(n), \mathbf{a}_2(n), \ldots, \mathbf{a}_l(n)))_{n\geq 0}$, where

$$\begin{aligned} \delta'((q_1, q_2, \ldots, q_l), c) &= (\delta_1(q_1, c), \delta_2(q_2, c), \ldots, \delta_l(q_l, c)), \\ \tau'((q_1, q_2, \ldots, q_l)) &= f((\tau_1(q_1), \tau_2(q_2), \ldots, \tau_l(q_l))), \end{aligned}$$

for all $q_i \in Q_i$ and $c \in \Sigma_k$. □

The class of automatic sequences is also closed under the so called right-shifts.

Theorem 24. *Let* $\mathbf{u} = (u_n)_{n\geq 0}$ *be a* k*-automatic sequence. Then so is* $\mathbf{v} = (v_n)_{n\geq 0}$ *defined by*

$$v_n = \begin{cases} u_{n-1} & \text{if } n \geq 1, \\ a & \text{if } n = 0, \end{cases}$$

where $a \in \Sigma_k$.

Proof. Suppose that, for some $r \geq 1$, $K_k(\mathbf{u}) = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$ is the k-kernel of $\mathbf{u}$. Define $\mathbf{v}_i$ by the rule

$$\mathbf{v}_i(n) = \begin{cases} \mathbf{u}_i(n-1) & \text{if } n \geq 1, \\ a & \text{if } n = 0. \end{cases}$$

We prove that the k-kernel of $\mathbf{v}$ is finite by showing that

$$K_k(\mathbf{v}) \subset \{\mathbf{u}_1, \ldots, \mathbf{u}_r, \mathbf{v}_1, \ldots, \mathbf{v}_r\}.$$

Consider the sequence $(\mathbf{v}(k^e n + j))_{n\geq 0}$, where $e \geq 0$ and $0 \leq j < k^e$. If $j \geq 1$, then $(\mathbf{v}(k^e n + j))_{n\geq 0} = (\mathbf{u}(k^e n + j - 1))_{n\geq 0} = (\mathbf{u}_i(n))_{n\geq 0}$ for some i, $0 \leq i \leq r$. Suppose then that $j = 0$. For some i and for all $n \geq 0$, we have now $\mathbf{u}(k^e n + k^e - 1) = \mathbf{u}_i(n)$. Hence $\mathbf{u}_i(n-1) = \mathbf{u}(k^e(n-1) + k^e - 1) = \mathbf{u}(k^e n - 1)$ for all $n \geq 1$. It follows that

$$\begin{aligned} \mathbf{v}(k^e n) &= \begin{cases} \mathbf{u}(k^e n - 1) & \text{if } n \geq 1, \\ a & \text{if } n = 0 \end{cases} \\ &= \begin{cases} \mathbf{u}_i(n-1) & \text{if } n \geq 1, \\ a & \text{if } n = 0 \end{cases} \\ &= \mathbf{v}_i(n). \end{aligned}$$

Thus $K_k(\mathbf{v})$ is finite and the result follows from Theorem 17. □

We are ready to prove the closure properties mentioned by Cobham in [8]. We begin by defining *shifts* $\mathcal{S}^i$ of infinite sequences. If $\mathbf{u} = u_0 u_1 u_2 \cdots$, then the shifted sequence $\mathcal{S}(\mathbf{u})$ is the word $u_1 u_2 u_3 \cdots$. Similarly, for $i \geq 0$, $\mathcal{S}^i(\mathbf{u}) = u_i u_{i+1} u_{i+2} \cdots$. For $i < 0$, $\mathcal{S}^i_w(\mathbf{u}) = w\mathbf{u}$ for a word w of length i.

Theorem 25. *The class of* k*-automatic sequences is closed under shifts.*

Proof. Suppose that $\mathbf{u} = (u_n)_{n\geq 0}$ is a k-automatic sequence. For $i \geq 0$, $\mathcal{S}^i(\mathbf{u}) = (u_{n+i})_{n\geq 0}$ is k-automatic by Theorem 20. For $i < 0$, we apply Theorem 24 i times. □

We say that an infinite sequence $\mathbf{b}$ is obtained from a sequence $\mathbf{a}$ by *periodic deletion* if there exist integers $p, q, j_0, \ldots, j_{p-1}$ with $1 \leq p < q$ and $0 \leq j_0 < \cdots < j_{p-1} < q$ such that $\mathbf{b}(pn + i) = \mathbf{a}(qn + j_i)$ for all $0 \leq i \leq p - 1$ and for all $n \geq 0$.

Theorem 26. *The class of* k*-automatic sequences is closed under periodic deletion.*

Proof. Suppose that $\mathbf{a}$ is a k-automatic sequence. By Theorem 20, $(\mathbf{a}(qn+j_i))_{n\geq 0}$ is k-automatic for any $q \geq 0$ and $j_i \geq 0$. Define a sequence $\mathbf{b}$ by the rule $\mathbf{b}(pn+i) = \mathbf{a}(qi+j_i)$. Thus $(\mathbf{b}(pn+i))_{n\geq 0}$ is k-automatic for $0 \leq i \leq p-1$ and the sequence $\mathbf{b}$ is k-automatic by Theorem 21. □

The sequence $\mathbf{b}$ is obtained from $\mathbf{a}$ by *q-block substitution* if $\mathbf{b} = f(\mathbf{a})$, where f is a q-uniform morphism. It means that $\mathbf{b}(qi+j) = f(\mathbf{a}(i))(j)$ for $0 \leq j \leq q-1$. Hence Corollary 3 can be reformulated as follows:

Theorem 27. *Let $q \geq 1$. The class of k-automatic sequences is closed under q-block substitution.*

By *q-block compression* we mean the following. Suppose that $\mathbf{a}$ is a sequence over Δ and let q be a positive integer. We group the successive symbols of $\mathbf{a}$ into q-blocks starting from the left and consider these blocks as single symbols. This q-block compressed sequence is an infinite word over the alphabet Δ^q.

Theorem 28. *Let $q \geq 1$. The class of k-automatic sequences is closed under q-block compression.*

Proof. Suppose that $\mathbf{a}$ is a k-automatic sequence. By Theorem 20, the sequences $(\mathbf{a}(qn+i))_{n\geq 0}$ are k-automatic for $0 \leq i \leq q-1$. The result follows now by applying Theorem 23 to these sequences and to the function $f\colon \Delta\times\cdots\times\Delta \to \Delta^q$, defined by $f((a_0, a_1, \ldots, a_{q-1})) = a_0a_1\cdots a_{q-1}$. □

2.6 Subword complexity

As it was already mentioned in the introduction, complexity of a sequence is an interesting mathematical notion related to number theory. We formulate this more precisely in the following.

Definition 3. The *subword complexity function* p_w of a word w is a function on nonnegative integers $\mathbb{N}$ defined by $p_w(n) = \#L_n(w)$, i.e. $p_w(n)$ is the number of different factors of length n in w.

Ultimately periodic sequences, i.e. sequences of the form xy^ω, where $x, y \in \Sigma^*$, $y \neq \varepsilon$, are highly ordered and their complexity function is bounded. There are also sequences with maximal complexity, i.e. $p_w(n) = |\Sigma|^n$. Consider, for example, the Champernowne word $\mathbf{c}$ [23, A030190] obtained by catenating the binary expansions of all positive integers in order.

$$\mathbf{c} = 0\,1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,1\,1\,1\cdots$$

It is easy to see that $p_{\mathbf{c}}(n) = 2^n$.

In this subsection we show that there is a linear upper bound on the subword complexity of automatic sequences. We begin by describing the relationship between automatic and periodic sequences, which continues in the next section. First we give a characterization of ultimately periodic sequences by Hedlund and Morse [13].

Theorem 29. *Let* $\mathbf{w}$ *be an infinite word. The following are equivalent:*

- *(i)* $\mathbf{w}$ *is ultimately periodic,*
- *(ii)* $p_{\mathbf{w}}(n) = p_{\mathbf{w}}(n+1)$ *for some* n*,*
- *(iii)* $p_{\mathbf{w}}(n) < n + k - 1$ *for some* $n \geq 1$*, where* k *is the number of different letters appearing in* $\mathbf{w}$*,*
- *(iv)* $p_{\mathbf{w}}(n)$ *is bounded.*

Proof. Suppose first that $\mathbf{w}$ is of the form xy^ω. Every factor of length n is contained in the set of factors of length n starting from the first $|xy|$ letters of $\mathbf{w}$. Thus (i) implies (iv). The implication $(iv) \implies (iii)$ is clear. Suppose next that $p_{\mathbf{w}}(m) < m + k - 1$ for some $m \geq 1$ and $k = p_{\mathbf{w}}(1)$. We may suppose that $k \geq 1$; for otherwise, $p_{\mathbf{w}}(n) = 1$ for all $n \geq 1$. Since the function $p_{\mathbf{w}}(n)$ is nondecreasing, we must have $p_{\mathbf{w}}(n) = p_{\mathbf{w}}(n+1)$ for some $n < m$; otherwise $p_{\mathbf{w}}(m) \geq m + k - 1$. Thus we have proved $(iii) \implies (ii)$. Suppose now that $p_{\mathbf{w}}(n) = p_{\mathbf{w}}(n+1)$ for some n. This means that each factor of length n can be extended uniquely to a factor of length $n+1$. Consider a word v of length n that occurs at least twice in the word $\mathbf{w}$. Denote by u the word between these two occurrences, i.e. vuv is a factor of $\mathbf{w}$. Since the letters that follow v are uniquely determined by u, $vuvu$ must be a factor of $\mathbf{w}$. Repeating this argument we see that $(vu)^\omega$ is a suffix of $\mathbf{w}$. Thus $\mathbf{w}$ is ultimately periodic, and so also $(ii) \implies (i)$. □

We may modify the above result to concern only *recurrent* factors, i.e. factors occurring infinitely often in $\mathbf{w}$. Let $r_{\mathbf{w}}(n)$ denote the number of distinct recurrent factors of length n in $\mathbf{w}$. Using similar argumentation as above we may prove

Theorem 30. *Theorem* 29 *holds for* $r_{\mathbf{w}}(n)$ *in place of* $p_{\mathbf{w}}(n)$*.*

Note that $p_{\mathbf{w}}(n)$ and $r_{\mathbf{w}}(n)$ can differ. Consider, for example, the infinite word $\mathbf{w} = baba^2ba^3b\cdots a^ib\cdots$. Words ba^ib are not recurrent. Thus $r_{\mathbf{w}}(n) < p_{\mathbf{w}}(n)$.

There is the following relation between automatic sequences and ultimately periodic sequences.

Theorem 31. *If* $(a_n)_{n\geq 0}$ *is an ultimately periodic sequence, then it is* k*-automatic for all* $k \geq 2$*.*

Proof. By Corollary 2 (page 23) it suffices to consider "purely" periodic sequences, i.e. sequences with $a_{tn+i} = a_i$ for some t and for all $0 \leq i \leq t-1$ and $n \geq 0$. We define a k-DFAO $M = (\Sigma_t,\, \Sigma_k,\, \delta,\, 0,\, \Delta,\, \tau)$, where

$$\begin{aligned} \delta(q,a) &= kq + a \pmod t \quad \text{and} \\ \tau(q) &= a_q \end{aligned}$$

for all $q \in \Sigma_t$ and $a \in \Sigma_k$. We claim that $\delta(0,w) = [w]_k \pmod t$ for all $w \in \Sigma_k^*$. If $|w| = 0$, this is clear. Assume then that the claim holds for all words of length

n. Suppose that $w = a_0 \cdots a_n$. Then

$$\begin{aligned}\delta(0, w) &= \delta(\delta(0, a_0 \cdots a_{n-1}), a_n) \\ &= \delta([a_0 \cdots a_{n-1}]_k \pmod t, a_n) \\ &= k[a_0 \cdots a_{n-1}]_k + a_n \pmod t \\ &= [w]_k \pmod t.\end{aligned}$$

Thus $\tau(\delta(0, w)) = a_{[w]_k \pmod t}$ and the result follows. □

The deep Cobham's theorem presented in the next section gives a kind of converse to the previous theorem. It states that if a sequence is both k-automatic and l-automatic, and k and l are multiplicatively independent, then the sequence is ultimately periodic. The following section is devoted to the proof of this beautiful result. The upper bound for the subword complexity of automatic sequences follows easily from the characterization using uniform morphisms and codings. First we prove a useful lemma.

Lemma 4. *Let* $\mathbf{u}$ *be an infinite word over* Σ*. Let* $\varphi\colon \Sigma^* \to \Delta^*$ *be a nonerasing morphism. Define* $\mathbf{v} = \varphi(\mathbf{u})$*. Then* $p_{\mathbf{v}}(n) \le W p_{\mathbf{u}}(n)$*, where* $W = \max\{|\varphi(a)| \mid a \in \Sigma\}$.

Proof. Let $\mathbf{v} = (v_n)_{n\ge 0}$ and $\mathbf{u} = (u_n)_{n\ge 0}$. Let x be a factor of $\mathbf{v}$ of length n, i.e. $x = v_i v_{i+1} \cdots v_{i+n-1}$ for some integer i. Define j to be the maximal index such that $|\varphi(u_0 u_1 \cdots u_{j-1})| \le i$. Since φ is nonerasing, x must be a factor of $\varphi(w)$, where $w = u_j \cdots u_{j+n-1}$. Actually, x is completely determined by the pair (w, k), where $k = i - |\varphi(u_0 \cdots u_{j-1})|$. More precisely, $x = (\varphi(w)(k))(\varphi(w)(k+1)) \cdots (\varphi(w)(k+n-1))$. Now $0 \le k \le |\varphi(u_j)| - 1 \le W - 1$. So, there cannot be more factors of $\mathbf{v}$ of length n than there are pairs (w, k). Thus $p_{\mathbf{v}}(n) \le W p_{\mathbf{u}}(n)$. □

Now we are ready to prove the upper bound. Recall that a k-automatic sequence can be generated by a k-uniform morphism under a coding.

Theorem 32. *Let* $\mathbf{v} = \tau(\varphi^\omega(a))$*, where* $\varphi\colon \Sigma^* \to \Sigma^*$ *is a* k*-uniform morphism and* $\tau\colon \Sigma^* \to \Delta^*$ *is a coding. Suppose that* $t = |\Sigma|$*. Then* $p_{\mathbf{v}}(n) \le k t^2 n$ *for all* $n \ge 1$.

Proof. Let $\mathbf{u} = \varphi^\omega(a) = u_0 u_1 u_2 \cdots$ and $n \ge 1$. Let r be the integer satisfying $k^{r-1} \le n < k^r$. For a fixed i, let $j = \lfloor \frac{i}{k^r} \rfloor$. Then $u_i u_{i+1} \cdots u_{i+n-1}$ is a factor of $u_{jk^r} \cdots u_{(j+1)k^r} \cdots u_{(j+2)k^r-1}$. Using Lemma 3 we conclude that $u_{jk^r} \cdots u_{(j+2)k^r-1} = \varphi^r(v_j v_{j+1})$. So, $u_i \cdots u_{i+n-1}$ is completely determined by $i \pmod{k^r}$, v_j and v_{j+1}. There are $k^r \cdot |\Sigma|^2$ different such triplets. Hence $p_{\mathbf{u}}(n) \le k^r t^2 \le k t^2 n$, where the last inequality comes from the definition of r. Since $\mathbf{v} = \tau(\mathbf{u})$ and $\max\{|\tau(a)| \mid a \in \Sigma\} = 1$, we have $p_{\mathbf{v}}(n) \le p_{\mathbf{u}}(n) \le k t^2 n$ by the previous lemma. □

To end this section, we shortly mention how these results are connected to transcendental number theory. Consider a base-k representation $0.\mathbf{v} = 0.v_0v_1v_2\cdots$ of a real number α. If the sequence $v_0v_1v_2\cdots$ is ultimately periodic, we know that α is rational. On the other hand, it is conjectured that the expansion in base k of an irrational algebraic number is *normal* in the sense that the expansion contains every block of digits of length n with a frequency asymptotic to $1/k^n$. The conjecture implies that numbers with low complexity expansion are either transcendental or rational. Since bounded subword complexity implies rationality by Theorem 29, the numbers with expansions of low and unbounded complexity should be transcendental. The first combinatorial result of this kind showed the transcendence of numbers, where $\mathbf{v}$ is Sturmian (for which $p_{\mathbf{v}}(n) = n + 1$) or Arnoux-Rauzy word (for which $p_{\mathbf{v}}(n) = 2n + 1$) [11]. Some other classes of sequences of complexity $2n+1$ were considered in [15]. The major improvement on this field is the resent result of Adamczewski et al. saying that the complexity function of the base k expansion of an irrational algebraic number satisfies $\liminf_{n\to\infty} \frac{p(n)}{n} = +\infty$ [1]. Thus, if a sequence $\mathbf{v}$ is nonperiodic and satisfies $p_{\mathbf{v}}(n) = \mathcal{O}(n)$, then $\alpha = 0.\mathbf{v}$ is transcendental. Consider then numbers $\alpha = 0.\mathbf{v}$, where $\mathbf{v}$ is an automatic sequence. Such numbers are called *automatic*. By the previous upper bound (Theorem 32), we have the following theorem, also mentioned in [1].

Theorem 33. *Algebraic irrational numbers cannot be automatic.*

3 Cobham's theorem

In this section we prove the famous Cobham's theorem [7]. It states that if a sequence is both k- and l-automatic for multiplicatively independent integers k and l, then the sequence is ultimately periodic. Recall that we proved in Theorem 31 that ultimately periodic sequences are k-automatic for all $k \geq 2$. This means that sequences can be divided into three categories:

(i) the sequences, which are automatic for all $k \geq 2$:
i.e. ultimately periodic sequences;
(ii) the sequences, which are automatic for powers of fixed k only:
e.g. the Thue-Morse sequence;
(iii) the nonautomatic sequences:
e.g. the characteristic sequence of squares.

Cobham's theorem is formulated for *k-automatic sets* instead of sequences. We first introduce some basic results on these automatic sets. Some number theoretical preliminaries are also considered. Syndetic sets are needed for the first part of the proof of the main theorem; see Section 3.3. This part of the main proof is also called *le Petit Théorème de Cobham* or Hansel's lemma. In Section 3.4 we consider k- and l-stable equivalence relations, which are also needed for the proof of Cobham's theorem presented in the last subsection. This proof is mainly due to Hansel [12], who first managed to simplify the original complex proof of Cobham. At present, some other reasonable proofs are known. We mention especially the proof of Michaux and Villemaire [18], which is based on logic via Büchi's theorem. In 1977, Semenov generalized Cobham's theorem to subsets M of $\mathbb{N}^m$, $m \geq 1$ [28]. We also mention the elegant proof of Muchnik [22] for the multi-dimensional case; see [5] for the English version.

3.1 Multiplicative independence

In this subsection we prove that the set of quotients $\{k^p/l^q \mid p, q \geq 0\}$ of multiplicatively independent integers k and l is dense in the set of positive real numbers. This will be proved using number theoretical results of Dirichlet and Kronecker. We begin by the definitions.

Definition 4. Two integers k and l are said to be *multiplicatively dependent* if there exists two integers $r, s \geq 1$ such that $k^r = l^s$. Otherwise, k and l are said to be *multiplicatively independent.*

Definition 5. A set S is *dense* in an interval I of reals if every open subinterval of I contains an element of S.

Next we prove the well-known Dirichlet's approximation theorem.

Theorem 34. (Dirichlet) *For all real numbers θ and positive integers N, there exists a positive integer $n \leq N$ and an integer r such that*

$$|n\theta - r| < \frac{1}{N}.$$

Proof. Define $\{\theta\} = \theta - \lfloor\theta\rfloor$, the fractional part of the real number θ. Consider $N+1$ numbers $\{0\}, \{\theta\}, \{2\theta\}, \ldots, \{N\theta\}$ and the N intervals $[\frac{m}{N}, \frac{m+1}{N})$, where $0 \leq m < N$. By the pigeon hole principle, at least two of the numbers fall into the same interval. Suppose that they are $\{i\theta\}$ and $\{j\theta\}$ with $0 \leq i < j \leq N$. Then $(j-i)\theta = (\lfloor j\theta\rfloor - \lfloor i\theta\rfloor) + (\{j\theta\} - \{i\theta\})$. Choose $n = j - i$ and $r = \lfloor j\theta\rfloor - \lfloor i\theta\rfloor$. By the above construction we clearly have $n \leq N$ and $|n\theta - r| < \frac{1}{N}$. □

Using this result it is easy to prove Kronecker's approximation theorem in one dimension.

Theorem 35. (Kronecker) *Let θ be an irrational number. For all real numbers α and all $\epsilon > 0$, there exist integers c, d such that*

$$|c\theta - \alpha - d| < \epsilon.$$

Proof. By Dirichlet's result (Theorem 34), for all $\epsilon < 1$, there exist integers a, b satisfying $|a\theta - b| < \epsilon$. Since θ is irrational, $|a\theta - b| > 0$. Suppose first that $a\theta - b > 0$ and let k be the greatest integer satisfying $0 \leq k(a\theta - b) < 1$. Consider now the sequence of points $0, \{a\theta\}, \{2a\theta\}, \ldots, \{ka\theta\}, 1$. Since $\{ia\theta\} = ia\theta - ib$ for all $0 \leq i \leq k$, we have $\{ia\theta\} - \{(i-1)a\theta\} = a\theta - b < \epsilon$ for $i = 1, 2, \ldots, k$. Since $k(a\theta - b) < 1 < (k+1)(a\theta - b)$, we also have $1 - \{ka\theta\} < \epsilon$. Thus we have proved that

$$0 < \{a\theta\} < \{2a\theta\} < \cdots < \{ka\theta\} < 1 \tag{9}$$

and the distance between two adjacent points is less than ϵ. Hence there exists an integer $i \in [0, k]$ such that $|\{ia\theta\} - \{\alpha\}| < \epsilon$. In other words, $|ia\theta - \lfloor ia\theta\rfloor - \alpha + \lfloor\alpha\rfloor| < \epsilon$. The result is obtained by choosing $c = ia$ and $d = \lfloor ia\theta\rfloor - \lfloor\alpha\rfloor$.

The case $a\theta - b < 0$ is proved similarly. We have $\{ia\theta\} = 1 - i(b - a\theta)$ for all i satisfying $0 \leq i(b - a\theta) < 1$ and equation (9) is changed into the equation

$$1 > \{a\theta\} > \{2a\theta\} > \cdots > \{ka\theta\} > 0. \tag{10}$$

□

Note that Kronecker's theorem can be expressed by saying that the set of fractional parts $\{\,\{n\theta\} \mid n \geq 0\}$ is dense in the interval $[0, 1)$. Now we are ready to prove our theorem concerning multiplicatively independent integers.

Theorem 36. *If k and l are multiplicatively independent integers, then the set of quotients $\{k^p/l^q \mid p, q \geq 0\}$ is dense in the positive reals.*

Proof. Let x be a positive real number. Define $\theta = \log_l k = \frac{\log k}{\log l}$ and $\alpha = \frac{\log x}{\log l}$. By the assumption of multiplicative independence, we know that θ is irrational. Namely, if $\theta = s/r$ for some integers s and r, then $r \log k = s \log l$. Thus $k^r = l^s$, which is a contradiction. Now by Kronecker's Theorem, for any positive ϵ, there exist integers c and d satisfying $|c\theta - \alpha - d| < \epsilon$. Multiplying by $\log l$ we have

$$|c \log k - \log x - d \log l| < \epsilon \log l. \tag{11}$$

This means that $c \log k - d \log l$ belongs to the interval $(\log x - \epsilon \log l, \log x + \epsilon \log l)$. Thus

$$\frac{k^c}{l^d} \in (xl^{-\epsilon}, xl^{\epsilon}). \tag{12}$$

By taking ϵ sufficiently small, we can fit this interval inside any open interval containing x. Thus the set of quotients $\{k^p/l^q \mid p, q \geq 0\}$ is dense in the positive reals. □

3.2 Automatic sets

Cobham's theorem is formulated for automatic sets, not for automatic sequences. Here we give a short introduction to this related notion.

Definition 6. The set $S \subseteq \mathbb{N}$ is called *k-automatic* if the characteristic sequence $(\chi_S(n))_{n>0}$ of S is k-automatic.

By definition, a set S is k-automatic if and only if the language $0^*(S)_k = 0^*\{(s)_k \mid s \in S\}$ is regular. Thus there exists a k-DFA A accepting $0^*(S)_k$ by Theorem 2. In that case, we say that the set S is *accepted* by A. As an example of k-automatic sets we mention arithmetic progressions of integers. For an arithmetic progression S, the sequence $(\chi_S(n))_{n>0}$ is ultimately periodic and we may apply Theorem 31. The class of k-automatic sets is closed under many operations, e.g. intersection, union and complement. These follow immediately from Theorem 23. We also have the following result, which we will need in the proof of Cobham's theorem.

Theorem 37. *If S is a k-automatic set, then for any integers $a, b \geq 0$ the set $J_{a,b}(S) = \{\, n \in \mathbb{N} \mid an + b \in S\}$ is also k-automatic.*

Proof. This follows easily from the closure properties of k-automatic sequences (Theorems 25 and 26). Namely, the sequence $(\chi_{J_{a,b}(S)}(n))_{n>0}$ is obtained from the sequence $\mathbf{u} = (\chi_S(n))_{n>0}$ by shifting it b times to the left and then deleting all the letters except the digits $\mathbf{u}(n)$ with n divisible by a. Since $(\chi_S(n))_{n>0}$ is k-automatic, the result follows from the above mentioned closure properties. □

We note that the previous theorem could also be proved directly by constructing the generating automaton of the sequence $(\chi_{J_{a,b}(S)}(n))_{n>0}$. By Theorem 11, we may assume that the inputs are read starting from the least significant bit. The desired automaton just simulates the behavior of the generating automaton of the sequence $(\chi_S(n))_{n>0}$ with an input $(an+b)_k$, which we can obtain, digit by digit, with an automaton performing the school algorithm of multiplication and addition.

3.3 Right dense and syndetic sets

Throughout this subsection $k, l \geq 2$ are multiplicatively independent integers. Assume that X is a set of integers which is both k- and l-automatic. Our aim is to prove that such a set is syndetic, which means that the distance of consecutive elements is bounded by a constant. This result is known as *le Petit Théorème de Cobham* (cf. [25]) or Hansel's lemma (cf. [19]). The proof presented here consists of several lemmas describing how a presentation of an arbitrary integer in either base k or l can be extended to present a number in X by adding extra digits in the end. The number of digits needed can be specified in advance. This phenomenon is related to the notion of right dense sets. The proof is based on the original proof of Cobham [7], since some of the later proofs of Cobham's little theorem are somewhat inadequate (cf. [16]). For different proofs, see also [19, 26]. We start with the definitions of right dense and syndetic sets.

Definition 7. A set $X \subseteq \Sigma^*$ is called *right dense* if for any word $u \in \Sigma^*$ there exists a word $v \in \Sigma^*$ such that uv belongs to the set X.

In other words, every word appears as a prefix of at least one word in X.

Definition 8. A set $X \subseteq \mathbb{N}$ is called *d-syndetic* if $X \cap [n, n+d) \neq \emptyset$ for all sufficiently large integers $n \geq 0$. A set is *syndetic* if it is d-syndetic for some d.

Next we show how these notions are related to each other.

Lemma 5. *If $X \subseteq \mathbb{N}$ is a syndetic set, then $0^*(X)_k$ is right dense.*

Proof. Assume that X is d-syndetic. Suppose that for all $n \geq n_0$ we have $X \cap [n, n+d) \neq \emptyset$. Choose an integer p such that $k^p \geq n_0 + d$. Then, for all $n \geq 0$, the interval $[nk^p, (n+1)k^p)$ contains an element of X. Hence there exists t such that $nk^p + t \in X$ and, for any $w \in \Sigma_k^*$ such that $[w]_k = n$, we have a word $u \in \Sigma_k^p$ such that $[u]_k = t$ and $wu \in 0^*(X)_k$. Thus $0^*(X)_k$ is right dense. □

Note that the converse is not true. If $0^*(X)_k$ is right dense, it does not mean that X is syndetic, not even in the case where X is k-automatic. Consider, for example, the set of integers with even number of digits in their normalized (beginning zeros ignored) base-k representation. This set $X = \{x_1 < x_2 < \cdots < x_n < \cdots\}$ is clearly k-automatic, but the distance between two consecutive elements can be arbitrarily large, i.e. $\sup(x_{n+1} - x_n) = \infty$. However, automatic sets are related to right dense sets. This will be formulated more precisely in the following lemmas. Multiplicatively independent integers play an important role in these results. Recall especially Theorem 36.

Lemma 6. *Suppose that $k, l \geq 2$ are multiplicatively independent integers. Let X be an infinite k-automatic set. Then for any integers c and d such that $d \geq c \geq 0$ and for any word $x \in \Sigma_l^*$ there exists a word $y \in \Sigma_l^*$ such that $xy \in 0^*(X)_l$ and $|y| \equiv c \pmod{d}$.*

Proof. Since X is k-automatic, the language $(X)_k$ is regular. Since X is also infinite, it contains arbitrarily long words. Thus, by pumping lemma (Lemma 1), there exist words t, u and v in Σ_k^* such that $tu^*v \subseteq (X)_k$. Let x be a word in Σ_l^*. Our aim is to find a word $y \in \Sigma_l^*$ such that $xy \in 0^*(X)_l$ and $|y| = dq + c$ for some integer q. For this purpose we use numbers $[tu^pv]_k$, $p \geq 0$, which by previous considerations belong to X. More precisely, we want to find $y \in \Sigma_l^*$ such that, for some $p \geq 0$,

$$[xy]_l = [tu^pv]_k \in X.$$

Thus we want to find an integer j, $0 \leq j < l^{dq+c}$ such that $[x]_l l^{dq+c} + j = [tu^pv]_k$. In other words, the following inequality must hold:

$$[x]_l l^{dq+c} \leq [tu^pv]_k < ([x]_l + 1)l^{dq+c}.$$

This can be written in the form

$$[x]_l l^{dq+c} \leq [v]_k + [u]_k \left(\frac{k^{gp+h} - k^h}{k^g - 1} \right) + [t]_k k^{gp+h} < ([x]_l + 1)l^{dq+c}, \qquad (13)$$

where $h = |v|$ and $g = |u|$. Now we make a common mathematical trick and reformulate the inequality by adding and subtracting suitable constants. Our aim is to divide inequality (13) into two parts so that it is possible to find a solution for one part by the density properties of quotients of multiplicatively independent integers and the other part is satisfied trivially. First, we divide equation (13) by l^{dq+c} and get

$$\left([x]_l + \frac{1}{4}\right) - \frac{1}{4} \leq \frac{[v]_k + [u]_k \left(\frac{k^{gp+h} - k^h}{k^g - 1} \right) + [t]_k k^{gp+h}}{l^{dq+c}} < \left([x]_l + 1 - \frac{1}{4}\right) + \frac{1}{4}.$$

The part depending on multiplicatively independent integers l^d and k^g is

$$[x]_l + \frac{1}{4} \leq \frac{k^{gp}}{l^{dq}} \frac{k^h}{l^c} \left(\frac{[u]_k}{k^g - 1} + [t]_k \right) < [x]_l + \frac{3}{4}. \qquad (14)$$

To simplify the notation, let $\frac{k^h}{l^c} \left(\frac{[u]_k}{k^g - 1} + [t]_k \right) = C$. The previous inequality has a solution, since by Theorem 36, there are arbitrarily large integers p and q such that

$$\frac{[x]_l + \frac{1}{4}}{C} \leq \frac{k^{gp}}{l^{dq}} < \frac{[x]_l + \frac{3}{4}}{C}.$$

Also, for q large enough, we trivially have

$$-\frac{1}{4} \leq \frac{[v]_k - [u]_k \frac{k^h}{k^g - 1}}{l^{dq+c}} < \frac{1}{4}. \qquad (15)$$

Adding inequalities (14) and (15) together we finally obtain the desired result. □

As a corollary of the previous lemma we have:

Corollary 4. *Suppose that $k, l \geq 2$ are multiplicatively independent integers. Let X be an infinite k-automatic set. Then the set $0^*(X)_l$ is right dense.*

Next we consider a k-automaton accepting an l-automatic set.

Lemma 7. *Suppose that $k, l \geq 2$ are multiplicatively independent integers. Let X be both k- and l-automatic set, and let q be any state in a k-DFA $A = (Q, \Sigma_k, \delta, q_0, F)$ accepting X. Then the set $X_q = \{n \mid \delta(q_0, (n)_k) = q\}$ is both k- and l-automatic.*

Proof. We may assume that A is minimal. Replacing the set of accepting states F of A by $\{q\}$ gives a k-automaton accepting $X_q = \{n \mid \delta(q_0, (n)_k) = q\}$. Now we have to show that X_q is also l-automatic. Let q' be any state of A different from q. Thus, by minimality, there exists a word $z \in \Sigma_k^*$ such that only one of the states $\delta(q, z)$ and $\delta(q', z)$ is an accepting state. If $\delta(q, z)$ is an accepting state, then define $T(q') = \{n \in \mathbb{N} \mid nk^{|z|} + [z]_k \in X\}$; otherwise, $T(q') = \{n \in \mathbb{N} \mid nk^{|z|} + [z]_k \notin X\}$. In both cases, by Theorem 37, $T(q')$ is l-automatic, since X is. Suppose now that $\delta(q_0, x) = q$. Then $\delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(q, z)$. Since $[xz]_k = [x]_k k^{|z|} + [z]_k$, we see that $[x]_k \in T(q')$ in either cases. Similarly, if $\delta(q_0, x) = q'$, then $[x]_k \notin T(q')$. Define now $T = \bigcap_{q' \neq q} T(q')$, where the intersection is over finitely many states of A different from q. As an intersection of l-automatic sets T is l-automatic and $[x]_k \in T$ if and only if $\delta(q_0, x) = q$. Hence the set $\{[x]_k \mid \delta(q_0, ([x]_k)_k) = q\}$ is l-automatic and the lemma follows. □

Before proving Lemma 9, which in fact implies Hansel's lemma, we still need a couple of definitions and a small number theoretical lemma.

Definition 9. Let $A = (Q, \Sigma_k, \delta, q_0, F)$ be a DFA. A state $q \in Q$ is *recurrent* if $\delta(q, x) = q$ for some $x \in \Sigma_k^+$. It is *properly recurrent* if also $\delta(q_0, y) = q$ for some $y \in \Sigma_k^* \backslash 0^*$.

Lemma 8. *Suppose that $\gcd(p_1, p_2, \ldots, p_k) = 1$, where $k \geq 2$. Then there exists an integer N such that any $n \geq N$ can be represented as a sum $\sum_{j=1}^{k} a_j p_j$, where the integers a_i are nonnegative.*

Proof. We prove the claim by induction on k. Suppose first that $k = 2$, i.e. $\gcd(p_1, p_2) = 1$. Then for all i there exists an integer $m \in \{0, 1, \ldots, p_1 - 1\}$ such that $mp_2 \equiv i \pmod{p_1}$. Otherwise, there exists integers m and m' such that $0 \leq m' < m \leq p_1 - 1$ and $mp_2 - m'p_2 \equiv 0 \pmod{p_1}$. Since $\gcd(p_1, p_2) = 1$, this implies that p_1 divides $m - m'$. But this is impossible, since $m - m' \leq p_1 - 1$. Now, if $n \geq (p_1 - 1)p_2$, then $n = a_1 p_1 + a_2 p_2$, where $a_2 \in \{0, 1, \ldots, p_1 - 1\}$ is chosen so that $a_2 p_2 \equiv n \pmod{p_1}$ and $a_1 = (n - a_2 p_2)/p_1$ is clearly nonnegative.

Suppose now that the claim holds for $k = l - 1$. We prove it for $k = l$. Now $\gcd(p_1, p_2, \ldots, p_l) = \gcd(p_1, \gcd(p_2, \ldots, p_l)) = 1$. Suppose that $\gcd(p_2, \ldots, p_l) = d$. Thus $\gcd(p_2/d, \ldots, p_l/d) = 1$. By the induction hypothesis, if $d = 1$ the case is clear. Otherwise there exists an integer N' such that for

any $n \geq N'$, $n = \sum_{j=2}^{l} b_j(p_j/d)$, where all b_i are nonnegative integers. Hence

$$nd = \sum_{j=2}^{l} b_j p_j. \tag{16}$$

Define now s to be the smallest integer satisfying $sp_1 \geq N'$. Using the same argumentation as above, we see that for all i, there exists integer $m \in \{sp_1, sp_1 + 1, \ldots, (s+1)p_1 - 1\}$ such that $md \equiv i \pmod{p_1}$. Suppose now that $n \geq ((s+1)p_1 - 1)d$. Then $n = a_1 p_1 + a_2 d$, where a_2 is an integer in the set $\{sp_1, sp_1 + 1, \ldots, (s+1)p_1 - 1\}$ satisfying $a_2 d \equiv n \pmod{p_1}$ and $a_1 = (n - a_2 d)/p_1$. Because of the conditions on the size of n and a_2, we see that a_1 is a nonnegative integer and $a_2 d$ can be presented in the form of equation (16). □

Lemma 9. *Suppose that $k, l \geq 2$ are multiplicatively independent integers. Let $A = (Q, \Sigma_k, \delta, q_0, F)$ be a k-automaton accepting an infinite l-automatic set X. Then there exists a constant N such that for any integer $n \geq N$, for any properly recurrent state $q \in Q$, and for any word $x \in \Sigma_k^*$, there exists $z \in \Sigma_k^*$ for which $|z| = n$ and $\delta(q_0, xz) = q$.*

Proof. Let $X_q = \{n \,|\, \delta(q_0, (n)_k) = q\}$ for a properly recurrent state $q \in Q$. Then X_q is infinite and, by Lemma 7, it is both k- and l-automatic. Suppose that $\delta(q_0, x_1) = q$ and denote

$$d_q = \gcd\{|y| \,|\, \delta(q, y) = q\}.$$

We apply Lemma 6 to the infinite l-automatic set X_q with $d = d_q$ and $c = 1$. Thus there exists $x_2 \in \Sigma_k^*$ such that $\delta(q_0, x_1 x_2) = q$ and $|x_2| \equiv 1 \pmod{d_q}$. On the other hand, $\delta(q, x_2) = q$, which implies that d_q divides $|x_2|$, i.e. $|x_2| \equiv 0 \pmod{d_q}$. This means that $d_q = 1$. Thus there exists a finite subset $\{|y_1|, \ldots, |y_k|\}$ of the set $\{|y| \,|\, \delta(q, y) = y\}$ such that $\gcd(|y_1|, \ldots, |y_k|) = 1$. It follows from Lemma 8 that there exists also a constant N_q such that any $n \geq N_q$ can be written in the form $\sum_{j=1}^{k} a_j |y_j|$. Thus, for any $n \geq N_q$, there is a word y for which $|y| = n$ and $\delta(q, y) = q$. Namely we may choose $y = y_1^{a_1} \cdots y_k^{a_k}$. Choose now $N = |Q| + \max(N_q)$, where the maximum is taken over the set of properly recurrent states of Q.

Let x be an arbitrary word in Σ_k^* and q an arbitrary properly recurrent state of Q. Again, Lemma 7 and Lemma 6 show the existence of a word z_1 such that $\delta(q_0, xz_1) = q$. By using the pumping lemma (Lemma 1) we may assume that $|z_1| \leq |Q|$. Then, for $n \geq N$, we have $N_q \leq N - |Q| \leq n - |z_1|$. By the above considerations there exists $z_2 \in \Sigma_k^*$ such that $\delta(q, z_2) = q$ and $|z_2| = n - |z_1|$. Thus, for $z = z_1 z_2$, we have $|z| = n$ and $\delta(q_0, xz) = q$. □

As a corollary of the previous lemma we finally have

Theorem 38. (Hansel) *Suppose that $k, l \geq 2$ are multiplicatively independent integers. If an infinite set of integers is both k- and l-automatic, then it is syndetic.*

Proof. Let X be an infinite set of integers which is both k- and l-automatic and let $A = (Q, \Sigma_k, \delta, q_0, F)$ be a k-automaton accepting $0^*(X)_k$. Since X is infinite, there exists a properly recurrent accepting state $q \in Q$. By Lemma 9, there is a constant $N = k^p$ such that for any word $x \in \Sigma_k^*$, there exists a word y of length N with $\delta(q_0, xy) = q \in F$. Thus $xy \in 0^*(X)_k$. Suppose that $x = (n)_k$. Then $[xy]_k = nk^p + [y]_k \in X$. This means that for any integer n there exists an integer t, $0 \leq t < k^p$, such that $nk^p + t \in X$. Hence X is $2k^p$-syndetic, since the distance between two consecutive elements of X must be less than $2k^p$. □

As a final result on syndetic sets we prove a technical lemma needed in the sequel.

Lemma 10. *Let X be a d-syndetic set of integers. Then for all positive integers K, L, h and each real $a > 0$ such that $K < L < K + a$, there exist $x \in X$ and an integer y such that $yL \leq xK + h \leq yL + da$.*

Proof. Let r be the smallest integer such that $rK + h < rL$. Then for all integers $i \geq 1$, we have $(r - i)L \leq (r - i)K + h$ by the minimality of r and also

$$(r - i)K + h = rK + h - iK < rL - iK < rL - i(L - a) = (r - i)L + ia.$$

Especially, for $1 \leq i \leq d$, we have

$$(r - i)L \leq (r - i)K + h < (r - i)L + da. \tag{17}$$

Adding jKL to (17) we obtain

$$(jK + r - i)L \leq (jL + r - i)K + h < (jK + r - i)L + da$$

for all i, $1 \leq i \leq d$. Since X is d-syndetic, for sufficiently large j and for some i with $1 \leq i \leq d$ the number $(jL+r-i)$ belongs to X. Thus choosing $x = jL+r-i$ and $y = jK + r - i$ completes the proof. □

3.4 Equivalence relations

We consider now equivalence relations needed in the theorems that follow. We deal with relations on the set on natural numbers as well as on the set Σ_k^*. Namely, any equivalence relation $\sim$ on Σ_k^* induces an equivalence relations on $\mathbb{N}$. Let $x, y \in \mathbb{N}$. Using base-k representations of integers we define

$$x \sim_k y \quad \Longleftrightarrow \quad (x)_k \sim (y)_k.$$

A notion corresponding to right-invariance is called k-stability.

Definition 10. An equivalence relation $\sim$ on $\mathbb{N}$ is *k-stable* if

$$x \sim y \Longrightarrow xk^j + t \sim yk^j + t$$

for all $j \geq 0$ and $0 \leq t < k^j$.

Lemma 11. *Let $\sim$ be an equivalence relation on Σ_k^*. The equivalence relation $\sim_k$ is k-stable if and only if $\sim$ is right invariant.*

Proof. Suppose first that $\sim$ is right invariant. Let $x \sim_k y$. Then $(x)_k \sim (y)_k$. Now $(xk^j + t)_k = (x)_k z$ and $(yk^j + t)_k = (y)_k z$, where $z \in \Sigma^j$. Since $\sim$ is right-invariant, $(x)_k z \sim (y)_k z$. Hence $xk^j + t \sim_k yk^j + t$. The converse statement can be proved similarly. □

Recall the equivalence relation introduced in Section 1.4. The Myhill-Nerode equivalence related to arbitrary language L is defined by

$$u \sim_L v \iff u^{-1}L = v^{-1}L.$$

Denote the equivalence relation of natural numbers obtained from the Myhill-Nerode equivalence relation by $\sim_{L,k}$. More precisely,

$$x \sim_{L,k} y \iff (x)_k \sim_L (y)_k.$$

Note that the relation $\sim_{L,k}$ is k-stable according to the previous lemma and right-invariance of $\sim_L$.

We give a short note on the quotient sets, on which the Myhill-Nerode relation is based. A regular language L can be characterized by the finiteness of the set $\{u^{-1}L \mid u \in \Sigma^*\}$ (Theorem 12). This is actually true also for the set $\{Lu^{-1} \mid u \in \Sigma^*\}$ and it is proved as Theorem III.8.1 in [10].

Lemma 12. *Let $L \subseteq \Sigma^*$. The set $\{u^{-1}L \mid u \in \Sigma^*\}$ is finite if and only if $\{Lu^{-1} \mid u \in \Sigma^*\}$ is finite.*

Proof. Suppose first that $\{Lu^{-1} \mid u \in \Sigma^*\}$ is finite. This implies that also $\{(u^R)^{-1}L^R \mid u^R \in \Sigma^*\}$ is finite, since $(Lu^{-1})^R = \{w^R \mid wu \in L\} = (u^R)^{-1}L^R$. Therefore, by Theorem 12, L^R is regular. Hence L is regular by Theorem 6. Using again Theorem 12 we see that also the set $\{u^{-1}L \mid u \in \Sigma^*\}$ is finite. The converse statement is true by symmetry. □

3.5 Proof of Cobham's theorem

Theorem 39. *Suppose that $k, l \geq 2$ are multiplicatively independent integers. If a set of integers X is both k- and l-automatic, then its characteristic sequence is ultimately periodic.*

Proof. If X is finite, then the characteristic sequence consists of zeros after a finite prefix. Thus we may assume that X is infinite. Suppose that the characteristic sequence of X is

$$\mathbf{c} = c_0 c_1 c_2 \cdots .$$

Before going into the details of the proof we describe shortly the idea in it. In the sequel we modify $\mathbf{c}$ with the equivalence relations of the previous chapter.

The idea is to construct an ultimately periodic sequence $\mathbf{v}$ so that its periodicity implies the periodicity of $\mathbf{c}$. The proof that the modified sequence is ultimately periodic is based on Theorem 30 dealing with the number of recurrent factors.

Let us first construct $\mathbf{v}$. Since X is k-automatic, the set $(X)_k$ is a regular language. According to Theorem 12 the relation $\sim_{(X)_k}$ is of finite index and $(X)_k$ is a union of some equivalence classes of the relation. Thus also $\sim_{(X)_k,k}$ is of finite index and X is a union of some equivalence classes of $\sim_{(X)_k,k}$. This means that we may define an infinite word $\mathbf{v} = v_0v_1v_2\cdots$, where v_n is the equivalence class of n with respect to the relation $\sim_{(X)_k,k}$. Since X is a union of some of the equivalence classes of $\sim_{(X)_k,k}$, we conclude that if $\mathbf{v}$ is ultimately periodic, so is also $\mathbf{c}$.

Next we show that there exists an integer m such that the number of recurrent factors of length m of $\mathbf{v}$ is bounded by m. The periodicity of $\mathbf{v}$ follows then from Theorem 30. For the proof we need to modify $\mathbf{v}$ by an l-stable equivalence relation which is a refinement of $\sim_{(X)_k,k}$. First we show that an equivalence class of $\sim_{(X)_k,k}$ is both k- and l-automatic. Suppose that E is an equivalence class of $\sim_{(X)_k,k}$ and let $x \in E$. We prove that

$$E = \bigcap_{x\in E_{jt}} E_{jt} \setminus \bigcup_{x\notin E_{jt}} E_{jt}, \tag{18}$$

where

$$E_{jt} = \{y \in \mathbb{N} \mid yk^j + t \in X\}$$

for all $j \geq 0$ and $0 \leq t < k^j$. Namely, if $x \sim_{(X)_k,k} y$ then for all $j \geq 0$ and $0 \leq t < k^j$ we have $x \in E_{jt}$ if and only if $y \in E_{jt}$. Thus $y \in \bigcap_{x\in E_{jt}} E_{jt} \setminus \bigcup_{x\notin E_{jt}} E_{jt}$. Conversely, suppose that y belongs to the right side of (18). Let z be an arbitrary word in Σ_k^*. Then

$$(y)_kz \in (X)_k \iff y \in E_{|z|,[z]_k} \iff x \in E_{|z|,[z]_k} \iff (x)_kz \in (X)_k.$$

Thus $(y)_k \sim_{(X)_k} (x)_k$, which implies that $y \sim_{(X)_k,k} x$. Hence $y \in E$ and identity (18) is hereby proved. Consider then the sets E_{jt} more closely. They are both k- and l-automatic by Theorem 37. Let now $u \in \Sigma_k^*$ be a word such that $|u| = j$ and $[u]_k = t$. We have

$$(E_{jt})_k = (X)_ku^{-1}.$$

Combining Theorem 12 and Lemma 12 we know that there exists only finitely many sets $(X)_ku^{-1}$. Hence there is only a finite number of sets $E_{jt} = [(X)_ku^{-1}]_k$. Now E is a finite boolean combination of l-automatic sets and therefore itself l-automatic. Similarly we conclude that E is also k-automatic.

Next we make an l-stable refinement of $\sim_{(X)_k,k}$. As mentioned earlier there is only a finite number N of equivalence classes E_i of $\sim_{(X)_k,k}$. Each of these classes E_i, $0 \leq i \leq N$, is l-automatic, which means that each $(E_i)_l$ is regular. By Theorem 12, there exists a finite index right-invariant equivalence relation of $(E_i)_l$. Using Lemma 11 we obtain an l-stable equivalence relation of E_i such that

it partitions the set E_i into a finite number N_i of equivalence classes E_{ij}. These sets E_{ij} with $0 \leq i \leq N$ and $0 \leq j \leq N_i$ partition X and define an equivalence relation $\sim_l$ which is clearly a refinement of $\sim_{(X)_k,k}$. Let $\mathbf{u}$ be the infinite sequence $\mathbf{u} = u_0u_1u_2\cdots$, where u_n is the equivalence class of n with respect to the relation $\sim_l$.

Now consider a recurrent factor $w = w_1w_2$ of length two of $\mathbf{v}$. Since all equivalence classes of $\sim_{(X)_k,k}$ are k- and l-automatic, the set $S_{w_1} = \{n \in \mathbb{N} \,|\, v_n = w_1\}$ is both k- and l-automatic. Using Theorem 37 we also conclude that $\{n \in \mathbb{N} \,|\, v_{n+1} = w_2\} = \{n \in \mathbb{N} \,|\, n+1 \in S_{w_2}\}$ is both k- and l-automatic. Thus so is their intersection $S_w = \{n \in \mathbb{N} \,|\, v_n = w_1, v_{n+1} = w_2\}$. We may then use Hansel's lemma (Theorem 38) to conclude that S_w is syndetic for any recurrent factor w. Hence there exists a positive integer d such that the distance between two occurrences of any recurrent subword of length 2 is at most d.

The final part of the proof is quite technical. The idea in it is the following. With the help of the syndetic result above we show that, for some positive integers p and q, each recurrent factor of $\mathbf{v}$ of fixed length m satisfying $m < l^q < 2k^p$ occurs inside some l^q-block and not very far from the beginning of the block. We count the number of l^q-blocks of $\mathbf{v}$ using the l-stability related to $\mathbf{u}$. This enables us to estimate the number of recurrent factors of length m precisely enough.

We showed earlier that the number of equivalence classes of $\sim_l$ is finite. Let c be this number and recall the definition of d from above. Then choose a real number ϵ such that

$$0 < \epsilon < 1 \quad \text{and} \quad \frac{c\epsilon}{1-\epsilon} < \frac{1}{2}. \tag{19}$$

By Theorem 36, we can find integers $p, q > 0$ such that

$$1 < \frac{l^q}{k^p} < 1 + \frac{\epsilon}{d}. \tag{20}$$

In order to simplify the notation, we denote $L = l^q$, $K = k^p$ and $m = \lfloor K(1-\epsilon) \rfloor$. We claim that for any recurrent factor w of length m of $\mathbf{v}$, there exists an integer y such that

$$v_{yL}v_{yL+1}\cdots v_{(y+1)L-1} = swt, \tag{21}$$

where $|s| \leq \epsilon K$.

To see this, we use the pigeon hole principle to conclude that a word w of length at most K must appear infinitely often at the same position in factors of the form $v_{xK}v_{xK+1}\cdots v_{(x+2)K-1}$. Since the equivalence relation $\sim_{(X)_k,k}$ is k-stable, we note that $v_{xK}v_{xK+1}\cdots v_{(x+2)K-1}$ is completely determined by the factor v_xv_{x+1}. Namely, if $v_x = v_{x'}$, then $x \sim_{(X)_k,k} x'$ and $xK + t \sim_{(X)_k,k} x'K + t$ for any $0 \leq t < K$. Thus $v_{xK+t} = v_{x'K+t}$ for any $0 \leq t < K$. By the definition of d, every recurrent factor of length 2 of $\mathbf{v}$ has a second occurrence at a distance at most d. So there exists an infinite set $X = \{x_1 < x_2 < \cdots < x_n < \cdots\}$ satisfying $x_{n+1} - x_n \leq d$ and

$$v_{x_nK}v_{x_nK+1}\cdots v_{(x_n+2)K} = w'ww''.$$

Let $h = |w'|$. Applying Lemma 10 with $a = \epsilon K/d$ to the d-syndetic set X we find integers x_n and y_n such that

$$y_n L \leq x_n K + h \leq y_n L + ad = y_n L + \epsilon K.$$

Thus we have a situation as in Figure 8, since also

$$x_n K + h + m \leq y_n L + K\epsilon + K(1-\epsilon) = y_n L + K < (y_n + 1)L.$$

Hence equation (21) is satisfied with $y = y_n$ and $|s| \leq \epsilon K$.

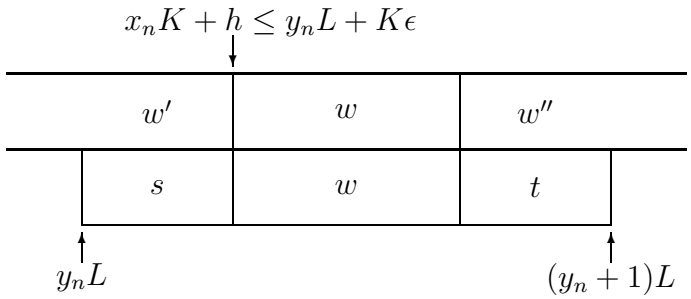


Figure 8: Illustration related to the proof

Now consider the number of factors of the form $v_{y_n L} v_{y_n L+1} \cdots v_{(y_n+1)L-1}$. It must be bounded by the number of factors of the form $u_{y_n L} u_{y_n L+1} \cdots u_{(y_n+1)L-1}$, since $\sim_l$ is a refinement of $\sim_{(X)_k,k}$. But $\sim_l$ is also l-stable, so we conclude as above that a word of the form $u_{y_n L} u_{y_n L+1} \cdots u_{(y_n+1)L-1}$ is entirely determined by u_{y_n}, i.e. by the equivalence class of y_n. Thus the number of recurrent factors $v_{y_n L} v_{y_n L+1} \cdots v_{(y_n+1)L-1}$ is bounded by c and the word w may start from any of the ϵK first letters of such a word. Hence the number of recurrent factors of length m of $\mathbf{v}$ is at most equal to

$$(\epsilon K)c \leq \frac{1}{2}K(1-\epsilon) \leq \frac{1}{2}(m+1) \leq m,$$

where the first inequality comes from equation (19). Thus, by Theorem 30, $\mathbf{v}$ is ultimately periodic. □

As a corollary, we finally get

Theorem 40. *Let $l, k \geq 2$ be multiplicatively independent integers, and suppose that the sequence $\mathbf{s} = (s_n)_{n\geq 0}$ is both k- and l-automatic. Then $\mathbf{s}$ is ultimately periodic.*

Proof. We may suppose that $\mathbf{s} \in \Delta^* = \Sigma_e^*$ for some integer $e \geq 2$. For each $a \in \Sigma_e$, consider the set $S_a = \{n \in \mathbb{N} \mid s_n = a\}$. Since $\mathbf{s}$ is both k- and l-automatic, so is each set S_a by Theorem 16. From Cobham's theorem it now follows that the characteristic sequence $\mathbf{w}_a$ of S_a is ultimately periodic. Suppose that $\mathbf{w}_a = x_a(y_a)^\omega$ for each $a \in \Sigma_e$. Let $c = \max\{|x_a| \mid a \in \Sigma_e\}$ and let u_a be the prefix of $\mathbf{w}_a$ of length c. We may then write $\mathbf{w}_a = u_a(y'_a)^\omega$. Let d be the least common multiple of the lengths $|y'_a|$. Suppose that $v_a = (y'_a)^{d/|y'_a|}$. Now $\mathbf{w}_a = u_a(v_a)^\omega$, where $|v_a| = d$. Thus $\mathbf{s}$ is ultimately periodic, since it is of the form uv^ω, where $|v| = d$, the letters $v(i)$ satisfy

$$v(i) \;=\; \sum_{a \in \Sigma_e} a v_a(i)$$

for $i = 0, 1, \ldots, d-1$ and u is the prefix of length c of $\mathbf{s}$. □

4 Shuffles of automatic sequences

In this section we consider shuffles of automatic sequences from the algorithmic point of view. We have already proved some closure properties concerning subsequences in Theorems 20 and 21. They imply that automatic sequences are closed under so called regular shuffles. Now our aim is to construct algorithms for the calculation of generating morphisms and codings of the shuffles in order to make these objects of discrete mathematics more concrete and pragmatic. This calculation of the morphisms and codings can, of course, be accomplished using the different stages of the constructive proofs presented in the previous sections, but the idea here is to make it in a more straightforward manner. For example, no reversing of automata, which in general is quite a complex operation, is needed in the algorithms. As an application we consider the construction of a generating automaton for Schröder numbers modulo 3. A short complex analysis and some generalizations are given in the end.

4.1 Definitions

Denote by $\mathbf{u}_{an+b} = (u_{an+b})_{n\geq 0}$ the subsequence of $\mathbf{u}$ for integers $a, b \geq 0$. We define *regular m-shuffles* of sequences $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$ with respect to a given rule α, where $\alpha = r_0 r_1 \cdots r_{N-1}$ is a word over Σ_m and $|\alpha|_i > 0$ for $i = 0, 1, \ldots, m-1$. Let also $\alpha_i = r_0 r_1 \cdots r_{i-1}$ for $i = 1, 2, \ldots, N$ and let $\alpha_0 = \varepsilon$. A sequence $\mathbf{u}$ is the regular shuffle of $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$ by the *rule* α, if

$$\mathbf{u}_{Nn+i} = \mathbf{u}^{(r_i)}_{|\alpha|_{r_i} n + |\alpha_i|_{r_i}}$$

for $i = 0, 1, \ldots, N-1$. For example, suppose that $m = 3$ and $\alpha = 01120$. This means that the shuffle $\mathbf{u}$ can be defined by blocks of length $N = |\alpha| = 5$. Namely,

$$\mathbf{u} = u_0^{(0)} u_0^{(1)} u_1^{(1)} u_0^{(2)} u_1^{(0)} u_2^{(0)} u_2^{(1)} u_3^{(1)} u_1^{(2)} u_3^{(0)} \cdots$$

From now on, we restrict ourselves to automatic sequences.

Theorem 41. *Each regular m-shuffle of k-automatic sequences is k-automatic. Conversely, if a regular m-shuffle of sequences $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m-1)}$ is k-automatic, then $\mathbf{u}^{(i)}$ is k-automatic for each $i = 0, 1, \ldots, m-1$.*

Proof. Let the sequences $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$ be k-automatic. Then by Theorem 20 the sequences $\mathbf{u}_{Nn+i} = \mathbf{u}^{(r_i)}_{|\alpha|_{r_i} n + |\alpha_i|_{r_i}}$ are k-automatic for $i = 0, 1, \ldots, N-1$. Thus applying Theorem 21 with $a = N$ we conclude that a regular m-shuffle of k-automatic sequences is k-automatic. For the converse, let $\mathbf{u}$ be k-automatic. By the definition of the shuffle, each sequence $\mathbf{u}^{(i)}_{|\alpha|_i n + j}$, where $i = 0, \ldots, m-1$ and $j = 0, \ldots, |\alpha|_i - 1$, is a subsequence of $\mathbf{u}$ of the form $(u_{an+b})_{n\geq 0}$ for some $a, b \geq 0$, and therefore k-automatic by Theorem 20. Thus every sequence $\mathbf{u}^{(i)}$ is k-automatic by Theorem 21 with $a = |\alpha|_i$. □

Note that an m-shuffle of k-automatic sequences is a k-uniformly morphic sequence by Theorem 18. This means that there exist a morphism $\varphi\colon \Gamma^* \to \Gamma^*$ and a coding $\tau\colon \Gamma^* \to \Delta^*$ such that $\mathbf{u} = \tau(\varphi^\omega(q_0))$ for a letter q_0 in a suitable alphabet Γ. Our aim is to find a simple algorithm for the calculation of such a triplet (τ, φ, q_0). As an input the algorithm gets triplets $(\tau_i, \varphi_i, q_{0i})$ generating the sequences $\mathbf{u}^{(i)} = \tau_i(\varphi_i^\omega(q_{0i}))$ which should be shuffled. We divide the examination into three parts starting from some special cases.

4.2 Perfect shuffles

We say that $\mathbf{u}$ is the *perfect m-shuffle* of sequences $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$ if

$$\mathbf{u}_{mn+i} = \mathbf{u}_n^{(i)}$$

for $i = 0, 1, \ldots, m-1$. Thus it is a regular m-shuffle with the rule $\alpha = 01\cdots m-1$. For example, if $m = 3$ then $\mathbf{u} = u_0^{(0)}u_0^{(1)}u_0^{(2)}u_1^{(0)}u_1^{(1)}u_1^{(2)}\cdots$.

In the following, we are going to present two algorithms. The first one gives a generating morphism and a coding of a perfect m-shuffle of k-automatic sequences in the special case $m = k$ and the second algorithm suits for any m.

4.2.1 Perfect k-shuffle of k-automatic sequences

Before presenting the algorithm, we fix some notation. Suppose that $\mathbf{u}^{(i)} = (u_n^{(i)})_{n\geq 0}$ is a k-automatic sequence for $i = 0, 1, \ldots, k-1$, and let $\mathbf{u}^{(i)}$ be generated by a k-DFAO $M_i = (\Gamma_i, \Sigma_k, \delta_i, q_{i0}, \Delta_i, \tau_i)$. By Theorem 18, we may also write $\mathbf{u}^{(i)} = \tau_i(\varphi_i^\omega(q_{i0}))$, where $\varphi_i\colon \Gamma_i^* \to \Gamma_i^*$ is a k-uniform morphism and $\tau_i\colon \Gamma_i^* \to \Delta_i^*$ is a coding. Suppose that $\Gamma_i = \{q_{i0}, q_{i1}, \ldots, q_{i(m_i-1)}\}$, where m_i is the size of the alphabet Γ_i. The output alphabet of the shuffle is $\Delta = \bigcup_{i=0}^{k-1}\Delta_i$. We also define the *cartesian product of k-uniform morphisms* $\varphi_i\colon \Gamma_i^* \to \Gamma_i^*$ for $i = 0, 1, \ldots, l$. It is a morphism

$$\varphi_0 \times \varphi_1 \times \cdots \times \varphi_l\colon (\Gamma_0 \times \Gamma_1 \times \cdots \times \Gamma_l)^* \to (\Gamma_0 \times \Gamma_1 \times \cdots \times \Gamma_l)^*$$

such that

$$\begin{array}{rl} & (\varphi_0 \times \varphi_1 \times \cdots \times \varphi_l)(a_0, a_1, \ldots, a_l) \\ = & (a_{00}, a_{10}, \ldots, a_{l0}) \cdots (a_{0(k-1)}, a_{1(k-1)}, \ldots, a_{l(k-1)}), \end{array}$$

where a_i is a letter in Γ_i and $\varphi_i(a_i) = a_{i0}a_{i1}\cdots a_{i(k-1)}$ for $i = 0, 1, \ldots, l$. Now we are ready to give the desired algorithm.

PERFECT_SHUFFLE_1

INPUT: $(\tau_0, \varphi_0, q_{00}), \ldots, (\tau_{k-1}, \varphi_{k-1}, q_{(k-1)0})$

1. For each $i = 0, 1, \ldots, k-1$:

   Let $\overline{\Gamma}_i = \{(a, b) \mid a \in \Gamma_i,\, b = \varphi_i(a)(i)\}$ and $\Gamma'_i = \Gamma_i \cup \overline{\Gamma}_i$. Define a morphism $\varphi'_i \colon \Gamma'^*_i \to \Gamma'^*_i$ by the image of each letter $q_{ij} \in \Gamma_i$ and $(q_{ij}, b) \in \overline{\Gamma}_i$: Suppose that $\varphi_i(q_{ij}) = a_0 \cdots a_{k-1}$. Then

   $$\begin{array}{lcl} \varphi'_i(q_{ij}) & = & a_0 \cdots a_{i-1}(q_{ij}, a_i)a_{i+1} \cdots a_{k-1}, \\ \varphi'_i((q_{ij}, a_i)) & = & \varphi'_i(a_i). \end{array}$$

2. Form the cartesian product

   $$\varphi = \varphi'_0 \times \cdots \times \varphi'_{(k-1)}.$$

3. Suppose that $(b_0, \ldots, b_{(k-1)}) \in \Gamma'_0 \times \cdots \times \Gamma'_{k-1}$. Define

   $$\tau((b_0, \ldots, b_{k-1})) = \tau_i(q_{ij}),$$

   if there exists a unique index i such that $b_i = (q_{ij}, b) \in \overline{\Gamma}_i$ for some $q_{ij} \in \Gamma_i$, (and $\tau((b_0, \ldots, b_{k-1})) = \tau_0(b_0)$, otherwise.)

4. Define $q_0 = (q_{00}, \ldots, q_{(k-1)0})$.

OUTPUT: (τ, φ, q_0)

We prove that the previous algorithm produces a generating morphism and a coding of the shuffle $\mathbf{u}$.

Theorem 42. *Suppose that* $\mathbf{u}^{(i)} = (u_n^{(i)})_{n\geq 0} = \tau_i(\varphi_i^\omega(q_{i0}))$ *for* $i = 0, \ldots, k-1$, *where* $\varphi_i \colon \Gamma_i^* \to \Gamma_i^*$ *is a* k*-uniform morphism and* $\tau_i \colon \Gamma_i^* \to \Delta_i^*$ *is a coding. Let* $\mathbf{u} = (u_n)_{n\geq 0}$ *be the perfect shuffle of sequences* $\mathbf{u}^{(i)}$. *Then the algorithm* PERFECT_SHUFFLE_1 *with input* $(\tau_0, \varphi_0, q_{00}), \cdots, (\tau_{k-1}, \varphi_{k-1}, q_{(k-1)0})$ *gives a triplet* (τ, φ, q_0) *such that* $\mathbf{u} = \tau(\varphi^\omega(q_0))$.

Proof. We prove the theorem using the connection between automatic and uniformly morphic sequences (Theorem 18). Instead of generating morphisms of $\mathbf{u}^{(i)}$ we consider corresponding k-DFAOs $M_i = (\Gamma_i, \Sigma_k, \delta_i, q_{0i}, \Delta_i, \tau_i)$. The idea is to construct a k-DFAO $M = (\Gamma, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $u_{kn+i} = \tau(\delta(q_0, (kn+i)_k)) = \tau_i(\delta_i(q_{0i}, (n)_k)) = u_n^{(i)}$. The Phase 1 of the algorithm corresponds to a construction of a modified automata $M'_i = (\Gamma'_i, \Sigma_k, \delta'_i, q_{i0}, \Delta_i, \tau'_i)$ for each $i = 0, 1, \ldots, k-1$. Now $\Gamma'_i = \Gamma_i \cup \overline{\Gamma}_i$, where $\overline{\Gamma}_i = \{(a, b) \mid a \in \Gamma_i,\, b = \delta_i(a, i)\}$. The transitions are

$$\begin{array}{lcl} \delta'_i(q_{ij}, l) & = & \delta_i(q_{ij}, l), \text{ if } l \neq i, \\ \delta'_i(q_{ij}, i) & = & (q_{ij}, \delta_i(q_{ij}, i)), \\ \delta'_i((q_{ij}, b), l) & = & \delta'_i(b, l) \end{array}$$

for $l = 0, 1, \ldots, k-1$.

First, we claim that if $\delta_i(q_{i0}, w_0 \cdots w_{l-1}) = a$ and $\delta_i(q_{i0}, w_0 \cdots w_{l-1}w_l) = b$ for any word $w_0 \cdots w_{l-1}w_l \in \Gamma_i^*$, then

$$\delta_i'(q_{i0}, w_0 \cdots w_{l-1}w_l) = \begin{cases} b & \text{if } w_l \neq i, \\ (a, b) & \text{if } w_l = i. \end{cases}$$

This can be easily proved by induction on l. The case $l = 0$ is clear. Otherwise, we have four different cases:

(a) $w_{l-1}, w_l \neq i$,
(b) $w_{l-1} \neq i, w_l = i$,
(c) $w_{l-1} = i, w_l \neq i$,
(d) $w_{l-1} = w_l = i$.

Suppose that the claim holds for $l = n$ with $\delta_i(q_{i0}, w_0 \cdots w_{n-1}) = a'$ and $\delta_i(q_{i0}, w_0 \cdots w_{n-1}w_n) = a$. Assume that $\delta_i(q_{i0}, w_0 \cdots w_n w_{n+1}) = b$. For example, if $w = w_0 \cdots w_{n-1}ii$, then using the induction assumption we see that $\delta_i'(q_{i0}, w) = \delta_i'(\delta_i'(q_{i0}, w_0 \cdots w_{n-1}i), i) = \delta_i'((a', a), i) = \delta_i'(a, i) = (a, \delta_i(a, i)) = (a, b)$. Thus the claim holds for $l = n + 1$ in the case (d). The other cases are proved similarly.

Now, if the final state of M_i is q_{ij} for input $(n)_k = w_0 w_1 \cdots w_l$ we claim that the final state of the modified automaton M_i' with input $(kn + i)_k = w_0 w_1 \cdots w_l i$ is $(q_{ij}, \delta_i(q_{ij}, i))$. This follows immediately from the previous claim. Phase 1 of the algorithm does not concern codings, they are inessential at this point (e.g. τ_i' could be chosen to be the identity mapping on Γ_i').

In the next two phases we make a union $M = (\Gamma, \Sigma_k, \delta, q_0, \Delta, \tau)$ of the automata M_i'. Define

$$\begin{aligned} \Gamma &= \Gamma_0' \times \cdots \times \Gamma_{k-1}', \\ \Delta &= \bigcup_{i=0}^{k-1} \Delta_i \quad \text{and} \\ \delta((b_0, \ldots, b_{k-1}), j) &= (\delta_0'(b_0, j), \ldots, \delta_{k-1}'(b_{k-1}, j)). \end{aligned}$$

Suppose now that $L_i' = \{n \in \mathbb{N} \mid \delta_i'(q_{i0}, (n)_k) \in \overline{\Gamma}_i\}$ for $i = 0, \ldots, k-1$. We claim that sets L_i' partition $\mathbb{N}$. Let $(n)_k = w_0 w_1 \cdots w_l$, where $w_l = n \pmod k$ is unique. By the definition of the modified automaton M_i' the only transitions to the states in $\overline{\Gamma}_i$ are of the form $\delta_i'(x, i)$ and also every $\delta_i'(x, i) \in \overline{\Gamma}_i$. Thus $i = n \pmod k$ is the only index such that with input $(n)_k$ the final state of M_i' is in $\overline{\Gamma}_i$. This proves that the latter part of the definition of τ in Phase 3 is never applied when generating the fixed point.

We have actually proven that in each letter $(b_0, \ldots, b_{k-1})$ of the fixed point of φ there exists only one index i such that $b_i \in \overline{\Gamma}_i$. We have also shown that, if $\delta_i(q_{i0}, (n)_k) = q_{ij}$, then $\delta_i'(q_{i0}, (kn + i)_k)) = (q_{ij}, \delta(q_{ij}, i))$. Thus we conclude

that, if $\delta_i(q_{i0}, (n)_k) = q_{ij}$, then

$$\begin{aligned}
&\tau(\delta(q_0, (kn+i)_k) \\
&= \tau(\delta'_0(q_{00}, (kn+i)_k), \ldots, \delta'_{(k-1)}(q_{(k-1)0}, (kn+i)_k)) \\
&= \tau(\ldots, \delta'_i(q_{i0}, (kn+i)_k), \ldots) \\
&= \tau(\ldots, (q_{ij}, \delta_i(q_{ij}, i)), \ldots) \\
&= \tau_i(q_{ij}) \\
&= \tau_i(\delta_i(q_{i0}, (n)_k)).
\end{aligned}$$

In other words, $(u_{kn+i})_{n\geq 0} = (u_n^{(i)})_{n\geq 0}$. □

As an example, we apply the algorithm to the Thue-Morse sequences $\mathbf{u}^{(0)} = \mathbf{u}^{(1)} = \mathbf{t}$. Using Theorem 18 to the automaton of Example 2 we see that $\mathbf{t}$ can be obtained by iterating the 2-uniform morphism $h\colon \Sigma_2 \to \Sigma_2$:

$$\begin{array}{ccc} 0 & \longmapsto & 01 \\ 1 & \longmapsto & 10 \end{array}$$

Now we simulate PERFECT_SHUFFLE_1 with input $(\mathrm{id}, h, 0), (\mathrm{id}, h, 0)$ to obtain the generating morphism of the perfect shuffle of two Thue-Morse sequences. After the first phase we have the following morphisms:

$$\begin{array}{lccl} \varphi'_0\colon & 0 & \mapsto & (0,0)1 \\ & 1 & \mapsto & (1,1)0 \\ & (0,0) & \mapsto & (0,0)1 \\ & (1,1) & \mapsto & (1,1)0 \end{array} \qquad \begin{array}{lccl} \varphi'_1\colon & 0 & \mapsto & 0(0,1) \\ & 1 & \mapsto & 1(1,0) \\ & (0,1) & \mapsto & 1(1,0) \\ & (1,0) & \mapsto & 0(0,1) \end{array}$$

In Phase 2 we can minimize the cartesian product by presenting only the images of those alphabets that are needed for the iteration of $\varphi^\omega((0,0))$. Thus we have φ :

$$\begin{array}{rcl} \big(0,0\big) & \mapsto & \big((0,0),0\big)\big(1,(0,1)\big) \\ \big((0,0),0\big) & \mapsto & \big((0,0),0\big)\big(1,(0,1)\big) \\ \big(1,(0,1)\big) & \mapsto & \big((1,1),1\big)\big(0,(1,0)\big) \\ \big((1,1),1\big) & \mapsto & \big((1,1),1\big)\big(0,(1,0)\big) \\ \big(0,(1,0)\big) & \mapsto & \big((0,0),0\big)\big(1,(0,1)\big) \end{array}$$

In order to simplify the notation we represent the morphism φ in another alphabet $\{A, B, C, D, E\}$:

$$\begin{array}{rccc} \varphi: & A & \mapsto & BC \\ & B & \mapsto & BC \\ & C & \mapsto & DE \\ & D & \mapsto & DE \\ & E & \mapsto & BC \end{array}$$

In Phase 3 we define the coding τ as follows:

$$\begin{array}{rccc} \tau: & A & \mapsto & 0 \\ & B & \mapsto & 0 \\ & C & \mapsto & 0 \\ & D & \mapsto & 1 \\ & E & \mapsto & 1 \end{array}$$

Thus we can now easily verify that

$$\varphi^\omega(A) = BCDEDEBCDEBCBCDEDEBCBCDEBCDEDE\cdots$$

and

$$\tau(\varphi^\omega(A)) = 00111100110000111100001100111 1\cdots,$$

which clearly is the perfect shuffle of two Thue-Morse sequences. Note that in this example the morphism φ obtained by the algorithm is not prolongable on the letter A, nevertheless the fixed point $\varphi^\omega(A)$ exists. This is guaranteed by Theorem 42.

As a final remark we note that for finding a generating morphism and a coding of a k-shuffle of k-automatic sequences we also could apply a related but different construction given by F. Durand [9, Prop. 3.1].

4.2.2 Perfect m-shuffle of k-automatic sequences

Suppose now that we have m k-automatic sequences $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$, where m may be different from k. We use the notation of the previous subsection. For general perfect shuffles we have the following algorithm:

PERFECT_SHUFFLE_2

INPUT: $(\tau_0, \varphi_0, q_{00}), \ldots, (\tau_{m-1}, \varphi_{m-1}, q_{(m-1)0})$

1. For each $i = 0, 1, \ldots, k-1$:

   Let $\Gamma_i' = \Gamma_i \times \{0, \ldots, m-1\}$. Define $\varphi_i'\colon \Gamma_i'^* \to \Gamma_i'^*$ as follows: Suppose that $\varphi_i(a) = a_0 a_1 \cdots a_{k-1}$. For $j = 0, 1, \ldots, m-1$

   $$\varphi_i'(a, j) = b_0 b_1 \cdots b_{k-1},$$

   where for $l = 0, 1, \ldots, k-1$

   $$b_l = (a_{\lfloor (kj+l)/m \rfloor}, kj + l \,(\mathrm{mod}\, m)).$$

2. Form the cartesian product $\varphi = \varphi'_0 \times \cdots \times \varphi'_{m-1}$

3. Suppose that $(b_0, \ldots, b_{m-1}) \in \Gamma'_0 \times \cdots \times \Gamma'_{m-1}$. Define

$$\tau((b_0, \ldots, b_{m-1})) = \begin{cases} \tau_i(a) & \text{if } b_i = (a, i) \text{ for exactly one } i, \\ \tau_0(b_0) & \text{otherwise.} \end{cases}$$

4. Define $q_0 = ((q_{00}, 0), \ldots, (q_{(m-1)0}, 0))$.

OUTPUT: (τ, φ, q_0)

We prove that the algorithm acts correctly.

Theorem 43. *Let* $\mathbf{u}^{(i)} = (u_n^{(i)})_{n\geq 0} = \tau_i(\varphi_i^\omega(q_0^{(i)}))$ *for* $i = 0, 1, \ldots, m-1$, *where* $\varphi_i \colon \Gamma_i^* \to \Gamma_i^*$ *is a* k*-uniform morphism and* $\tau_i \colon \Gamma_i^* \to \Delta_i^*$ *is a coding. Let* $\mathbf{u} = (u_n)_{n\geq 0}$ *be the perfect shuffle of the sequences* $\mathbf{u}^{(i)}$. *Then the algorithm* PERFECT_SHUFFLE_2 *with input* $(\tau_0, \varphi_0, q_{00}), \ldots, (\tau_{m-1}, \varphi_{m-1}, q_{(m-1)0})$ *gives a triplet* (τ, φ, q_0) *such that* $\mathbf{u} = \tau(\varphi^\omega(q_0))$.

Proof. Instead of generating morphisms of the sequences $\mathbf{u}^{(i)}$ we consider again corresponding k-DFAOs $M_i = (\Gamma_i, \Sigma_k, \delta_i, q_{0i}, \Delta_i, \tau_i)$. Suppose that $M'_i = (\Gamma'_i, \Sigma_k, \delta'_i, (q_{0i}, 0), \Delta_i, \tau'_i)$ is a k-DFAO such that

$$\delta'_i((a, j), l) = (\delta_i(a, \lfloor (kj + l)/m \rfloor), kj + l \,(\mathrm{mod}\, m)).$$

As in the proof of Theorem 42 the coding τ'_i is inessential. The automaton M'_i corresponds to the morphism φ'_i by Theorem 18.

We claim first that $\delta'_i((q_{i0}, 0), (mn+i)_k) = (\delta_i(q_{i0}, (n)_k), i)$. The new automaton simulates the school algorithm of division by m in base k. Consider two states (a, j) and (a', j') in Γ'_i and a transition $\delta_i((a, j), l)$ from the former state to the latter. The second component j of the state $(a, j) \in \Gamma'_i$ corresponds to the current remainder modulo m. In the school algorithm we get the next digit of the quotient by reading the new digit l of the dividend and calculating $\lfloor (kj + l)/m \rfloor$. The new remainder j' is $kj + l \mod m$, which is by definition the second component of (a', j'). The change of the first components corresponds the transition in M_i from a to the state a', when the digit $\lfloor (kj + l)/m \rfloor$ is read. Let $(N)_k$ be the input of M'_i. Thus in the first components of Γ'_i we simulate the original automaton M_i with input $(\lfloor N/m \rfloor)_k$. If $N = mn + i$, then the first component of the final state is the final state of M_i with input $(n)_k$, i.e. the state $\delta_i(q_{i0}, (n)_k)$, and the second component is i. Thus $\delta'_i((q_{i0}, 0), (mn + i)_k) = (\delta_i(q_{i0}, (n)_k), i)$.

As in Theorem 42 we form the union $M = (\Gamma, \Sigma_k, \delta, q_0, \Delta, \tau)$ of the automata

M_i'. This corresponds to Phases 2 and 3 of the algorithm. We have

$$\begin{aligned}
\Gamma &= \Gamma_0' \times \cdots \times \Gamma_{m-1}', \\
\Delta &= \bigcup_{i=0}^{k-1} \Delta_i, \\
\delta((b_0, \ldots, b_{m-1}), j) &= (\delta_0'(b_0, j), \ldots, \delta_{m-1}'(b_{m-1}, j)),
\end{aligned}$$

where $b_i \in \Gamma_i'$. The coding τ is defined as in Phase 3. As in Theorem 42 we show that the latter part of the definition of τ is never applied when generating the fixed point. Suppose that

$$L_i = \{n \in \mathbb{N} \mid \delta_i'((q_{i0}, 0), (n)_k) = (a, i) \text{ for some } a \in \Gamma_i\}$$

for $i = 0, 1, \ldots, m-1$. Since there exists a unique $i \in \{0, \ldots, m-1\}$ such that $n \equiv i \pmod m$, the sets L_i partition $\mathbb{N}$ and in each letter $(b_0, \ldots, b_{m-1})$ of the fixed point of φ there exists only one index i such that $b_i = (a, i)$ for some $a \in \Gamma_i$. Thus, by the previous calculations,

$$\begin{aligned}
&\tau(\delta(q_0, (mn+i)_k) \\
&= \tau(\delta_0'((q_{00}, 0), (mn+i)_k), \ldots, \delta_{m-1}'((q_{(m-1)0}, 0), (mn+i)_k)) \\
&= \tau(\ldots, \delta_i'((q_{i0}, 0), (mn+i)_k), \ldots) \\
&= \tau(\ldots, (\delta_i(q_{i0}, (n)_k), i), \ldots) \\
&= \tau_i(\delta_i(q_{i0}, (n)_k)).
\end{aligned}$$

Since by Theorem 18 the automaton M corresponds to the morphism φ and the coding τ with respect to the generation of the automatic sequence, the previous calculation implies that $\mathbf{u}_{mn+i} = \mathbf{u}_n^{(i)}$, where $\mathbf{u} = \tau(\varphi^\omega(q_0))$. Thus the algorithm gives the desired triplet. □

As an example, we apply the algorithm to the Thue-Morse sequence, the dual Thue-Morse sequence obtained by interchanging 0 and 1 in $\mathbf{t}$ and the sequence of 2's. So we have

$$\begin{aligned}
\mathbf{u}^{(0)} &= \mathbf{t} = h^\omega(0), \\
\mathbf{u}^{(1)} &= \overline{\mathbf{t}} = 10010110011010010110 \cdots = h^\omega(1) \quad \text{and} \\
\mathbf{u}^{(2)} &= 222 \cdots = \varphi_2^\omega(2),
\end{aligned}$$

where morphism h is like in the previous example and $\varphi_2(2) = 22$. In Phase 1 we get

$$
\begin{array}{lll}
\varphi_0: & (0,0)\mapsto(0,0)(0,1) & \varphi_1: \quad (1,0)\mapsto(1,0)(1,1) \\
& (0,1)\mapsto(0,2)(1,0) & \quad\ (1,1)\mapsto(1,2)(0,0) \\
& (0,2)\mapsto(1,1)(1,2) & \quad\ (1,2)\mapsto(0,1)(0,2) \\
& (1,0)\mapsto(1,0)(1,1) & \quad\ (0,0)\mapsto(0,0)(0,1) \\
& (1,1)\mapsto(1,2)(0,0) & \quad\ (0,1)\mapsto(0,2)(1,0) \\
& (1,2)\mapsto(0,1)(0,2) & \quad\ (0,2)\mapsto(1,1)(1,2)
\end{array}
\qquad
\begin{array}{ll}
\varphi_2: & (2,0)\mapsto(2,0)(2,1) \\
& (2,1)\mapsto(2,2)(2,0) \\
& (2,2)\mapsto(2,1)(2,2)
\end{array}
$$

For example, the last row of φ_1 is calculated in the following way. The number of sequences to be shuffled is 3 and we deal with 2-uniform sequences. Thus $m=3$, $k=2$ and $(a,j)=(0,2)$. Then $\varphi_1(a)=\varphi_1(0)=a_0a_1=01$. If $l=0$, then $kj+l=2\cdot 2+0=4$, $\lfloor (kj+l)/m\rfloor=\lfloor 4/3\rfloor=1$ and $kj+l \pmod 3 = 1$. The first letter is $b_0=(a_1,1)=(1,1)$. Similarly, with $l=1$, we have $kj+l=2\cdot 2+1=5$, $\lfloor (kj+l)/m\rfloor=\lfloor 5/3\rfloor=1$ and $kj+l \pmod 3 = 2$. The second letter is $b_1=(a_1,2)=(1,2)$.

We minimize the representation of the cartesian product of the morphisms of Phase 1 as before. Note that for the calculation of the fixed point $\mathbf{u}$ we need only cartesian products of the form $\big((b_0,j),\ldots,(b_{m-1},j)\big)$, where $j=0,\ldots,m-1$ is the same for all coordinates. The morphisms φ is

$$
\begin{array}{rcl}
\big((0,0),(1,0),(2,0)\big) & \mapsto & \big((0,0),(1,0),(2,0)\big)\big((0,1),(1,1),(2,1)\big) \\
\big((0,1),(1,1),(2,1)\big) & \mapsto & \big((0,2),(1,2),(2,2)\big)\big((1,0),(0,0),(2,0)\big) \\
\big((0,2),(1,2),(2,2)\big) & \mapsto & \big((1,1),(0,1),(2,1)\big)\big((1,2),(0,2),(2,2)\big) \\
\big((1,0),(0,0),(2,0)\big) & \mapsto & \big((1,0),(0,0),(2,0)\big)\big((1,1),(0,1),(2,1)\big) \\
\big((1,1),(0,1),(2,1)\big) & \mapsto & \big((1,2),(0,2),(2,2)\big)\big((0,0),(1,0),(2,0)\big) \\
\big((1,2),(0,2),(2,2)\big) & \mapsto & \big((0,1),(1,1),(2,1)\big)\big((0,2),(1,2),(2,2)\big)
\end{array}
$$

We represent the morphism in another alphabet $\{A,B,\ldots,F\}$ and in Phase 3 we define the coding τ.

$$
\begin{array}{lrcl}
\varphi: & A & \mapsto & AB \\
& B & \mapsto & CD \\
& C & \mapsto & EF \\
& D & \mapsto & DE \\
& E & \mapsto & FA \\
& F & \mapsto & BC
\end{array}
\qquad
\begin{array}{lrcl}
\tau: & A & \mapsto & 0 \\
& B & \mapsto & 1 \\
& C & \mapsto & 2 \\
& D & \mapsto & 1 \\
& E & \mapsto & 0 \\
& F & \mapsto & 2
\end{array}
$$

For example, let $E=\big((1,1),(0,1),(2,1)\big)$. Now $b_1=(0,1)$ is the unique letter such that $b_i=(a,i)$. Thus $\tau(E)=\tau_1(0)=\mathrm{id}(0)=0$.

Iterating the morphism φ we have

$$\varphi^\omega(A)=ABCDEFDEFABCDEFABCABCDEFDEFABC\cdots$$

and

$$\tau(\varphi^\omega(A))=012102102012102012012102102012\cdots,$$

which is the perfect shuffle of the sequences $\mathbf{t}, \overline{\mathbf{t}}$ and $\mathbf{u}_2$. Note that the algorithm PERFECT_SHUFFLE_2 does not always give the minimal representation of the generating morphism with respect to the number of letters. This is the case, for example, in the calculations of Section 4.4.

4.3 Regular shuffles

Consider now a general regular m-shuffle $\mathbf{u}$ of k-automatic sequences $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(m-1)}$ shuffled by a rule $\alpha = r_0 r_1 \cdots r_{N-1}$. In order to find a generating morphism and a coding for $\mathbf{u}$ we may use the algorithms of the previous subsection. The idea is to construct first a new set of sequences $\mathbf{v}^{(i)}$, $i = 0, \ldots, N-1$, so that the perfect shuffle $\mathbf{v}$ of $\mathbf{v}^{(i)}$, i.e. the N-shuffle by the rule $\alpha' = 01 \cdots N-1$, corresponds to $\mathbf{u}$. This means that

$$\mathbf{v}^{(i)} = \mathbf{v}_{Nn+i} = \mathbf{u}_{Nn+i} = \mathbf{u}^{(r_i)}_{|\alpha|_{r_i} n + |\alpha_i|_{r_i}}$$

for $i = 0, 1, \ldots, N-1$. In other words, we have to construct an algorithm which gives a triplet (τ', φ', q_0') for the subsequence of the form $\mathbf{u}^{(i)}_{an+b}$ when a triplet (τ, φ, q_0) for $\mathbf{u}^{(i)}_n$ is given as input. Using automata this means the following: Suppose that $A = (\Gamma, \Sigma_k, \delta, q_0, \Delta, \tau)$ is a k-DFAO, which generates $\mathbf{u}^{(i)}$. We have to construct a k-DFAO $M_{ab} = (\Gamma_{ab}, \Sigma_k, \delta_{ab}, q_{0ab}, \Delta, \tau_{ab})$ generating $\mathbf{u}^{(i)}_{an+b}$ such that for input $(n)_k$ it simulates the behavior of A with input $(an+b)_k$. By definition of automatic sequences we are obliged to read the input $(n)_k$ from left to right. Performing constant multiplication and addition starting from the most significant digit makes the algorithm somewhat complicated. By Theorem 11, we could construct first automaton M'_{ab} which uses constant multiplication and addition from right to left and then calculate automaton M'^R_{ab} for the reversed finite-state function. But in that case we possibly need an exponential number of states compared to the number of states of M_{ab} and consequently compared to the number of states of the original automaton A. Thus we would end up in complex calculations. In the following we try to do this task in a simpler and more straightforward manner.

Let us first consider the school algorithm for multiplication (Figure 9). Suppose that we have input $(n)_k = n_0 n_1 \cdots n_{l_n}$, constant $(a)_k = a_0 a_1 \cdots a_l$ and output $p_0 p_1 \cdots p_{l_n+l+1} \in \Sigma_k^*$ such that $[p_0 p_1 \cdots p_{l_n+l+1}]_k = p = na$. Indeed, we may need $l_n + l + 2$ digits to represent the product, since the greatest possible p is $(k^{l+1}-1)(k^{l_n+1}-1) = k^{l_n+l+2} - k^{l_n+1} - k^{l+1} + 1$. In order to keep our notation simple we allow zeros in the beginning of the output. In the school algorithm the idea is to divide the multiplication into *partial products* and then add these by associativity:

$$na = (\sum_{i=0}^{l_n} n_i k^{l_n - i}) \cdot a = \sum_{i=0}^{l_n} (n_i \cdot a) k^{l_n - i}.$$

In practice, this means that we calculate first the products $r_i = n_i \cdot a$. The base-k representations $r_{i0} r_{i1} \cdots r_{i(l+1)}$ of products r_i are called *multiplication rows*. Note

that $l+1$ is the greatest possible nonzero power of k in the representation of r_i, since the greatest value of r_i is $(k-1)(k^{l+1}-1) = k^{l+2}-k^{l+1}-k+1$. In the school algorithm we write these rows under each other so that all the digits r_{ij} where $i+j=k$ are in a same column. This corresponds to multiplication of the partial products r_i by k^{l_n-i}. Finally, we perform the addition using school algorithm with carries c_i. Thus

$$\begin{aligned} p_j &= c_j + \sum_{i+k=j} r_{ik} \pmod{k} \text{ and} \\ c_{j-1} &= \frac{1}{k}\Big(\Big(c_j + \sum_{i+k=j} r_{ik}\Big) - p_j\Big), \end{aligned}$$

for $i = l_n+l+1, \ldots, 1$, and $c_{l_n+l+1} = 0$. Note that carries are counted from right to left in this procedure.

$$\begin{array}{llllllll}
 & & & a_0 & \cdots & a_{l-1} & a_l \\
\times & & & n_0 & n_1 & \cdots & n_{l_n-1} & n_{l_n} \\
\hline
c_0 & c_1 & \cdots & c_l & c_{l+1} & \cdots & c_{l+l_n} & c_{l+l_n+1} \\
 & & & r_{l_n 0} & r_{l_n 1} & \cdots & r_{l_n l} & r_{l_n (l+1)} \\
 & & r_{(l_n-1)0} & r_{(l_n-1)1} & r_{(l_n-1)2} & \cdots & r_{(l_n-1)(l+1)} & \\
 & & \iddots & \iddots & \iddots & \iddots & \iddots & \\
 & r_{10} & & \cdots & r_{1l} & r_{1(l+1)} & & \\
+ & r_{00} & r_{01} & \cdots & r_{0(l+1)} & & & \\
\hline
p_0 & p_1 & & \cdots & p_{l+1} & p_{l+2} & \cdots & p_{l+l_n+1}
\end{array}$$

Figure 9: Multiplication by school algorithm

Now our idea is to modify this school algorithm represented in Figure 9 so that the calculations are performed from left to right. The problem in multiplication from left to right is the control of the carries. In order to minimize values of the carries we use *intermediate sums* s_i. We start calculating multiplication rows and intermediate sums from left to right, i.e. starting from the bottom in Figure 10. Intermediate sums

$$s_i = r_i + (s_{i-1} \pmod{k^{l+1}}) \cdot k$$

are represented by $c_i s_{i0} s_{i1} \cdots s_{i(l+1)} \in \Sigma_k^*$ for $i = 0, \ldots, l_n$. The initial value $s_{(-1)} = 0$. This corresponds to line $c_{(-1)} s_{(-1)0} s_{(-1)1} \cdots s_{(-1)(l+1)} = 0^{l+3}$.

Consider now the greatest value for s_i. Clearly we have

$$s_i \leq (k^{l+2} - k^{l+1} - k + 1) + (k^{l+1} - 1)k = 2k^{l+2} - k^{l+1} - 2k + 1.$$

This means that either $c_i = 0$ or $c_i = 1$. Define $c'_i = \lfloor (s_{i0} + c_{i+1} + c'_{i+1})/k \rfloor$ for $i = 1, \ldots, l_n - 1$ and $c'_{l_n} = 0$. Considering all possible values of c_{i+1} and s_{i0} it is

easy to prove by induction that $c'_i \leq 1$ for all i. Comparing now Figures 9 and 10 we see that the correct value for p_i, where $i = -1, \ldots, l_n - 1$, is $s_{i0} + c_{i+1} + c'_{i+1} \pmod k$ and, for $i = l_n, \ldots, l_n + l + 1$, $p_i = s_{(l_n)i}$. Note that there are a lot of extra zeros in Figure 10. Namely $l_n + l + 2$ digits are enough for the representation of the result p. Thus in addition to zeros $c_{(-1)}s_{(-1)0}s_{(-1)1}\cdots s_{(-1)(l+1)}$ we have $c_0 = c'_0 = p_{-1} = 0$. These zeros are needed in the sequel in the construction of an automaton based on this algorithm.

In other words, during the calculation of the intermediate sums we do not know the right values for c'_i. But after calculating the digits of the last intermediate sum s_{l_n} we can start to calculate carries c'_i and digits p_i from right to left. From another viewpoint, after the calculation of each s_i we could guess whether $c'_i = 0$ or $c'_i = 1$ and calculate the value of p_{i-1}. Thus the digits p_i could be calculated from left to right and in the end we could verify the correctness of our guesses. This algorithm seems to be nondeterministic because of the guesses, but actually it can be quite easily modified to a deterministic one. In addition, we can easily modify this algorithm to perform calculations of the form $an+b$, where $b = 0, 1, \ldots, a-1$. We just replace the definition of r_{l_n} with $r_{l_n} = n_{l_n} \cdot a + b$. Since $b \leq a - 1 \leq k^{l+1} - 2$, then $s_{l_n} \leq (2k^{l+2} - k^{l+1} - 2k + 1) + (k^{l+1} - 2) = 2k^{l+2} - 2k - 1$ and we have $c_{l_n} \leq 1$. Thus $c'_i \leq 1$ for all i and, as before, the result $p = na + b$ can contain only $l_n + l + 2$ digits.

$$
\begin{array}{llllllllll}
 & & & & a_0 & & \cdots & a_{l-1} & a_l & \\
\times & & & & n_0 & n_1 & \cdots & n_{l_n-1} & n_{l_n} & \\
\hline
 & & & \overset{c'_{l_n}}{c_{l_n}} & s_{l_n 0} & s_{l_n 1} & \cdots & s_{l_n l} & s_{l_n (l+1)} & \\
 & & & & r_{l_n 0} & r_{l_n 1} & \cdots & r_{l_n l} & r_{l_n (l+1)} & \\
 & & \overset{c'_{l_n-1}}{c_{l_n-1}} & s_{(l_n-1)0} & s_{(l_n-1)1} & s_{(l_n-1)2} & \cdots & s_{(l_n-1)(l+1)} & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
 & \overset{c'_1}{c_1} & s_{10} & & \cdots & s_{1l} & s_{1l} & s_{1(l+1)} & & \\
 & & r_{10} & & \cdots & r_{1(l-1)} & r_{1l} & r_{1(l+1)} & & \\
\overset{c'_0}{c_0} & s_{00} & s_{01} & & \cdots & s_{0l} & s_{0(l+1)} & & & \\
\hline
 & r_{00} & r_{01} & & \cdots & r_{0l} & r_{0(l+1)} & & & \\
+\ c_{(-1)} & s_{(-1)0} & s_{(-1)1} & s_{(-1)2} & & \cdots & s_{(-1)(l+1)} & & & \\
\hline
 & p_{-1} & p_0 & p_1 & & \cdots & p_l & p_{l+1} & \cdots & p_{l+l_n+1}
\end{array}
$$

Figure 10: Multiplication from left to right

Recall $A = (\Gamma, \Sigma_k, \delta, q_0, \Delta, \tau)$ and $M_{ab} = (\Gamma_{ab}, \Sigma_k, \delta_{ab}, q_{0ab}, \Delta, \tau_{ab})$. Next we define M_{ab} with the help of the multiplication algorithm described above. Note that this automaton is deterministic regardless of the guesses of the carries in the above formulation. Suppose that $[a_0 \cdots a_l]_k = a$, $[b_0 \cdots b_l]_k = b$ and $0 \leq b < a$.

The set of states of the automaton is

$$\Gamma_{ab} = \{(q, q', s) |\, q, q' \in \Gamma, s = s_0 s_1 \cdots s_l s_{l+1} \in \Sigma_k^{l+2}\}$$

and the initial state is $q_{0ab} = (q_0, q_0, 0^{l+2})$. The first two coordinates q and q' correspond to carries $c_i' = 0$ and $c_i' = 1$, respectively. Thus we will use the notations $q = q^{(0)}$ and $q' = q^{(1)}$ in the following.

Let x be an integer in $\{0, 1, \ldots, k-1\}$ and $s = s_0 s_1 \cdots s_{l+1}$. Define $\overline{c}\,\overline{s} = \overline{c}\overline{s}_0\overline{s}_1 \cdots \overline{s}_{l+1}$, $c_0 d_0$ and $c_1 d_1$ to be the unique words satisfying $[\overline{c}\overline{s}_0\overline{s}_1 \cdots \overline{s}_{l+1}]_k = [s_1 \cdots s_{l+1}]_k k + ax$, $[c_0 d_0]_k = s_0 + \overline{c} + 0$ and $[c_1 d_1]_k = s_0 + \overline{c} + 1$. Then the transition function is

$$\begin{aligned}
\delta_{ab}((q^{(0)}, q^{(1)}, s), x) &= (\overline{q}^{(0)}, \overline{q}^{(1)}, \overline{s}), \quad \text{where} \\
\overline{q}^{(0)} &= \begin{cases} \delta(q^{(0)}, d_0) & \text{if } c_0 = 0, \\ \delta(q^{(1)}, d_0) & \text{if } c_0 = 1, \end{cases} \\
\overline{q}^{(1)} &= \begin{cases} \delta(q^{(0)}, d_1) & \text{if } c_1 = 0, \\ \delta(q^{(1)}, d_1) & \text{if } c_1 = 1. \end{cases}
\end{aligned}$$

Suppose also that $c''s'' = c''s_0''s_1'' \cdots s_{l+1}''$ is the word satisfying

$$[c''s'']_k = [s]_k + b.$$

Then the output function is defined as follows:

$$\tau_{ab}((q^{(0)}, q^{(1)}, s)) = \begin{cases} \tau(\delta(q^{(0)}, s'')) & \text{if } c'' = 0, \\ \tau(\delta(q^{(1)}, s'')) & \text{if } c'' = 1. \end{cases}$$

Now it is easy to see that if the input of the automaton M_{ab} is $(n)_k$, then it simulates the original automaton A with input $(an + b)_k$. We just establish the connection between the definition of the automaton and the previous multiplication algorithm. The nondeterministic guesses of the carries are handled by using two state coordinates, $q^{(0)}$ and $q^{(1)}$, which correspond to values 0 and 1 of the carry c_i', respectively. Denote the values of the current state by overlined symbols and symbols with no overline correspond to the previous state. Now suppose that we know the correct value of the carry bit $\overline{c}'$. It determines the correct carry bit c' of the previous state. For example, if $\overline{c}' = 0$, then the correct digit of the product p is $(s_0 + \overline{c} + 0 \pmod k) = d_0$. In addition, we know the right value of c'. Namely, $c' = \lfloor (s_0 + \overline{c} + 0)/k \rfloor = c_0$. Hence we know, which one of the states $q^{(0)}$ and $q^{(1)}$ was the right guess in the previous state. It was $q^{(1)}$, if $c_0 = 1$, and we calculate $\overline{q}^{(0)}$ by the formula $\delta(q^{(1)}, d_0)$.

The third coordinate of the initial state $q_{0ab} = (q_0, q_0, 0^{l+2})$ corresponds to the digits $s_{(-1)0}s_{(-1)1} \cdots s_{(-1)(l+1)} = 0^{l+2}$ of Figure 10. Actually, $q_{0ab} = (q_0, q_0, 0^{l+2})$ is not the only possible choice for the initial state of M_{ab}, since the value of the second coordinate $q^{(1)}$ can be chosen freely. It does not affect the transitions. Namely, if $[\overline{c}\overline{s}]_k = [0^{l+2}]_k k + ax$, then clearly $\overline{c} = 0$. Therefore $s_0 + \overline{c} + 0 = 0$

and $s_0 + \overline{c} + 1 = 1$. Hence $c_0 = d_0 = c_1 = 0$ and $d_1 = 1$. The first transition is then $\delta_{ab}((q_0, q_0, 0^{l+2}), n_0) = (\delta(q_0, 0), \delta(q_0, 1), s_{00}s_{01}\cdots s_{0(l+1)})$. By the definition of automatic sequences we may assume that $\delta(q_0, 0) = q_0$. Thus $\delta_{ab}((q_0, q_0, 0^{l+2}), n_0) = (q_0, \delta(q_0, 1), s_{00}s_{01}\cdots s_{0(l+1)})$. The second coordinate $\delta(q_0, 1)$ is again meaningless. Namely, $c'_0 = 0$ in Figure 10, which means that the first coordinate is the correct guess. The meaning of this first transition is a kind of initialization of the simulation of A.

After reading the whole input we end up to some state of the automaton M_{ab}. In this final state we have already simulated the multiplication $a \cdot n$ except for the last $l + 2$ digits. These digits are $p_{l_n}\cdots p_{l_n+l+1} = s_{l_n 0}\cdots s_{l_n(l+1)}$. For the multiplication $a \cdot n$ the correctly guessed carry of the final state is 0, the correct state coordinate is $q^{(0)}$ and the correct simulation of the remaining digits is performed in the transition $\delta(q^{(0)}, s_{l_n 0}\cdots s_{l_n(l+1)})$. But we want also to perform the addition by b. The output function takes care of this using the same kind of idea as above. We define that $c''s'' = c''s''_0 s''_1 \cdots s''_{l+1}$ is the unique word satisfying $[c''s''_0 s''_1 \cdots s''_{l+1}]_k = [s]_k + b$. The only possible values for c'' are 0 and 1. If $c'' = 1$, it just means that $q^{(1)}$ is the correctly guessed state coordinate. Thus the effect of the remaining digits of the base k representations of $an + b$ is formulated as $\delta(q^{(1)}, s'')$. If $c'' = 0$, then the transition is $\delta(q^{(0)}, s'')$. Finally, the output function τ is used. We summarize this in terms of morphisms and codings.

SUBSEQUENCE

INPUT: (τ, φ, q_0) and $a_0\cdots a_l, b_0\cdots b_l \in \Sigma_k^{l+1}$, where $0 < [b_0\cdots b_l]_k = b < [a_0\cdots a_l]_k = a$

1. Let $\Gamma_{ab} = \{(q^{(0)}, q^{(1)}, s) |\, q^{(0)}, q^{(1)} \in \Gamma, s = s_0 s_1 \cdots s_l s_{l+1} \in \Sigma_k^{l+2}\}$ and $q_{0ab} = (q_0, q_0, 0^{l+2})$.

2. Define a morphism $\varphi_{ab}\colon \Gamma_{ab}^* \to \Gamma_{ab}^*$,

$$\varphi_{ab}(q^{(0)}, q^{(1)}, s_0 s_1 \cdots s_l s_{l+1}) = (q_0^{(0)}, q_0^{(1)}, \overline{s}_0)\cdots(q_{k-1}^{(0)}, q_{k-1}^{(1)}, \overline{s}_{(k-1)}),$$

where using the words $\overline{c}_i\overline{s}_i = \overline{c}_i\overline{s}_{i0}\overline{s}_{i1}\cdots\overline{s}_{i(l+1)}$, $c_{i0}d_{i0}$ and $c_{i1}d_{i1}$ satisfying

$$\begin{aligned}
[\overline{c}_i\overline{s}_i]_k &= [\overline{c}_i\overline{s}_{i0}\overline{s}_{i1}\cdots\overline{s}_{i(l+1)}]_k = [s_1\cdots s_{l+1}]_k k + ai,\\
[c_{i0}d_{i0}]_k &= s_0 + \overline{c}_i + 0 \quad \text{and}\\
[c_{i1}d_{i1}]_k &= s_0 + \overline{c}_i + 1
\end{aligned}$$

we have

$$\begin{aligned}
q_i^{(0)} &= \begin{cases} \varphi(q^{(0)})(d_{i0}) & \text{if } c_{i0} = 0,\\ \varphi(q^{(1)})(d_{i0}) & \text{if } c_{i0} = 1,\end{cases}\\
q_i^{(1)} &= \begin{cases} \varphi(q^{(0)})(d_{i1}) & \text{if } c_{i1} = 0,\\ \varphi(q^{(1)})(d_{i1}) & \text{if } c_{i1} = 1.\end{cases}
\end{aligned}$$

3. Define a coding $\tau_{ab}\colon \Gamma_{ab}^* \to \Delta^*$ as follows: Suppose that $c''s'' = c''s_0''s_1'' \cdots s_{l+1}''$ is the unique word satisfying $[c''s'']_k = [s]_k + b$. Then

$$\tau_{ab}((q^{(0)}, q^{(1)}, s) = \begin{cases} \tau(\varphi(\cdots\varphi(\varphi(q^{(0)})(s_0''))(s_1'')\cdots)(s_{l+1}'')) & \text{if } c''=0, \\ \tau(\varphi(\cdots\varphi(\varphi(q^{(1)})(s_0''))(s_1'')\cdots)(s_{l+1}'')) & \text{if } c''=1. \end{cases}$$

OUTPUT: $(\tau_{ab}, \varphi_{ab}, q_{0ab})$

By the above considerations we conclude

Theorem 44. *Suppose that* $\mathbf{u} = (u_n)_{n\geq 0} = \tau(\varphi^\omega(q_0))$*, where* $\varphi\colon \Gamma^* \to \Gamma^*$ *is a* k*-uniform morphism and* $\tau\colon \Gamma^* \to \Delta^*$ *is a coding. Let* $a_0\cdots a_l$ *and* $b_0\cdots b_l$ *be words over* Σ_k *such that* $0 < [b_0\cdots b_l]_k = b < [a_0\cdots a_l]_k = a$ *for some integers* a *and* b*. Then the algorithm* SUBSEQUENCE *with input* (τ, φ, q_0)*,* $a_0\cdots a_l$*,* $b_0\cdots b_l$ *gives a triplet* $(\tau_{ab}, \varphi_{ab}, q_{0ab})$ *such that* $\mathbf{u}_{an+b} = \tau_{ab}(\varphi^\omega(q_{0ab}))$*.*

As an example, we consider once again the Thue-Morse sequence, i.e. $(\tau, \varphi, q_0) = (h, \mathrm{id}, 0)$. Suppose that $a = 3 = [11]_2$ and $b = 1 = [01]_2$. In this case we may reduce the lengths of multiplication rows and intermediate sums by one, i.e. $l + 1 = 1$. The reason is that in base 2 the largest value of any multiplication rows is $1 \cdot a$. Thus only $|a|$ digits are needed. In order to simplify the notation, we denote $(q^{(0)}, q^{(1)}, s)$ by $[q^{(0)}q^{(1)}s_0\cdots s_l]_2$. For example, $(0, 1, 10) = [0110]_2 = 6$. Using the algorithm SUBSEQUENCE we have the following morphism φ_{ab} and coding τ_{ab}:

φ_{ab}:				τ_{ab}:			
	0	$\mapsto$	$4\cdot 7$		0	$\mapsto$	1
	1	$\mapsto$	$6\cdot 9$		1	$\mapsto$	1
	2	$\mapsto$	$8\cdot 11$		2	$\mapsto$	0
	3	$\mapsto$	$10\cdot 5$		3	$\mapsto$	0
	4	$\mapsto$	$4\cdot 7$		4	$\mapsto$	1
	5	$\mapsto$	$6\cdot 13$		5	$\mapsto$	1
	6	$\mapsto$	$12\cdot 15$		6	$\mapsto$	0
	7	$\mapsto$	$14\cdot 9$		7	$\mapsto$	1
	8	$\mapsto$	$8\cdot 11$		8	$\mapsto$	0
	9	$\mapsto$	$10\cdot 1$		9	$\mapsto$	0
	10	$\mapsto$	$0\cdot 3$		10	$\mapsto$	1
	11	$\mapsto$	$2\cdot 5$		11	$\mapsto$	0
	12	$\mapsto$	$8\cdot 11$		12	$\mapsto$	0
	13	$\mapsto$	$10\cdot 5$		13	$\mapsto$	0
	14	$\mapsto$	$4\cdot 7$		14	$\mapsto$	1
	15	$\mapsto$	$6\cdot 9$		15	$\mapsto$	1

For example, consider the state $(0, 0, 11) = 3$. We calculate

$$\begin{array}{lcllcll}
s_0s_1 & = & 11, & & & & \\
([s_1\cdots s_{l+1}]_k k + a\cdot 0)_2 & = & (1\cdot 2 + 3\cdot 0)_2 & = & 010 & = & \overline{c}_0\overline{s}_{00}\overline{s}_{01}, \\
(s_0 + \overline{c}_0 + 0)_k & = & (1 + 0 + 0)_2 & = & 01 & = & c_{00}d_{00}, \\
(s_0 + \overline{c}_0 + 1)_k & = & (1 + 0 + 1)_2 & = & 10 & = & c_{01}d_{01}, \\
([s_1\cdots s_{l+1}]_k k + a\cdot 1)_2 & = & (1\cdot 2 + 3\cdot 1)_2 & = & 101 & = & \overline{c}_1\overline{s}_{10}\overline{s}_{11}, \\
(s_0 + \overline{c}_1 + 0)_k & = & (1 + 1 + 0)_2 & = & 10 & = & c_{10}d_{10}, \\
(s_0 + \overline{c}_1 + 1)_k & = & (1 + 1 + 1)_2 & = & 11 & = & c_{11}d_{11}
\end{array}$$

and

$$\begin{array}{lllllll} q_0^{(0)} & = & \varphi(q^{(0)})(d_{00}) & = & h(0)(1) & = & 1, \quad \text{since } c_{00}=0, \\ q_0^{(1)} & = & \varphi(q^{(1)})(d_{01}) & = & h(0)(0) & = & 0, \quad \text{since } c_{01}=1, \\ q_1^{(0)} & = & \varphi(q^{(1)})(d_{10}) & = & h(0)(0) & = & 0, \quad \text{since } c_{10}=1, \\ q_1^{(1)} & = & \varphi(q^{(1)})(d_{11}) & = & h(0)(1) & = & 1, \quad \text{since } c_{11}=1. \end{array}$$

Thus

$$3 \;=\; (0,0,11) \;\mapsto\; (1,0,10)(0,1,01) \;=\; 10\cdot 5.$$

Now $([s]_k + b)_k = ([11]_2 + 1)_2 = (4)_2 = 100 = c''s'' = c''s_0''s_1''$. Since $c'' = 1$, the output is $\tau_{ab}(0,0,11) = \tau(\varphi(\varphi(q^{(1)})(s_0''))(s_1'')) = \mathrm{id}(h(h(0)(0))(0)) = 0$.

By using minimization (see Section 1.4) to the automata M_{ab} we get simpler morphisms and codings. The minimized form of the previous example is presented in the following.

$$\begin{array}{lllll} \varphi: & 0 & \mapsto & 0\cdot 1 \\ & 1 & \mapsto & 0\cdot 2 \\ & 2 & \mapsto & 1\cdot 3 \\ & 3 & \mapsto & 4\cdot 2 \\ & 4 & \mapsto & 5\cdot 3 \\ & 5 & \mapsto & 5\cdot 4 \end{array} \qquad \begin{array}{lll} \tau: & 0 & \mapsto & 1 \\ & 1 & \mapsto & 1 \\ & 2 & \mapsto & 0 \\ & 3 & \mapsto & 1 \\ & 4 & \mapsto & 0 \\ & 5 & \mapsto & 0 \end{array}$$

Iterating morphism φ we have

$$\varphi^\omega(0) = 0102011301020242010201130113531301 02\cdots$$

and

$$\tau(\varphi^\omega(0)) = 1110111111101000111011111111011111 10\cdots .$$

Comparing this sequence to the Thue-Morse sequence, we see that it is $\mathbf{t}_{3n+1}$.

$$\begin{aligned} \mathbf{t} \;=\;& 0\underline{1}10\underline{1}00\underline{1}10\underline{0}10\underline{1}10\underline{1}00\underline{1}01\underline{1}00\underline{1}10\underline{1}00\underline{1}10\underline{0}10\underline{1}10\underline{0} \\ & 11\underline{0}10\underline{0}10\underline{1}10\underline{1}00\underline{1}10\underline{0}10\underline{1}10\underline{1}00\underline{1}01\underline{1}00\underline{1}10\underline{1}00\underline{1}01\underline{1} \\ & 01\underline{0}01\underline{1}00\underline{1}01\underline{1}00\underline{1}10\underline{1}00\underline{1}10\underline{0}\cdots . \end{aligned}$$

To summarize, by combining the algorithm SUBSEQUENCE with the algorithm PERFECT_SHUFFLE_2, we are able to calculate triplets (τ, φ, q_0) for regular shuffles of k-automatic sequences with any given rule α.

4.4 Applying algorithms to Schröder numbers

In this section we consider Schröder numbers introduced in Example 4; see page 21. We show, how the automaton of Figure 7 is obtained. For the construction we use shuffling algorithm PERFECT_SHUFFLE_2.

Let us denote the sequence of Schröder numbers by $(S_n)_{n\geq 0}$. We have

$$(S_n)_{n\geq 0} = 1,\ 2,\ 6,\ 22,\ 90,\ 394,\ 1806,\ 8558,\ 41586,\ \ldots,$$

where the commas are only for clarification. This is the sequence A006318 in Sloane's On-Line Encyclopedia of Integer Sequences [23]. It satisfies the recurrence relation

$$S_{n+1} = 3S_n + \sum_{k=1}^{n-1} S_k S_{n-k}$$

for $n \geq 1$ with $S_0 = 1$ and $S_1 = 2$. Consider first the sequence $\mathbf{s}_{2n} = (S_{2n} \pmod 3)_{n\geq 0}$. It can be proved that S_n is divisible by 3 for every positive even integer n [30]. Thus

$$\mathbf{s}_{2n} = 1\,0\,0\,0\cdots$$

and it is clearly a fixed point $\varphi_0^\omega(1)$, where φ_0 is a 3-uniform morphism

$$\begin{aligned} 1 &\mapsto 100 \\ 0 &\mapsto 000. \end{aligned}$$

The residue modulo 3 of Schröder numbers with odd index gives a little more complicated sequence:

$$\mathbf{s}_{2n+1} = (S_{2n+1} \,(\mathrm{mod}\, 3))_{n\geq 0} = 2\,1\,1\,2\,1\,0\,0\,0\,1\cdots$$

We construct an automaton generating this sequence starting from the formula

$$S_{2n-1} \pmod 3 = \begin{cases} 0 & \text{if } 2 \in L_1((\lfloor m/3 \rfloor)_3), \\ 1 & \text{if } 2 \notin L_1((\lfloor m/3 \rfloor)_3) \text{ and } m \not\equiv 1 \pmod 3, \\ 2 & \text{if } 2 \notin L_1((\lfloor m/3 \rfloor)_3) \text{ and } m \equiv 1 \pmod 3, \end{cases}$$

for all $n \geq 1$ [33]. Recall that $L_1(w)$ denotes the set of the factors of w of length 1. This formula gives the automaton $M_{2n-1} = (Q,\, \Sigma_3,\, \delta,\, q_0,\, \Sigma_3,\, \tau)$ in Figure 11.

Next we convert this to an automaton $M_{2n+1} = (Q',\, \Sigma_3,\, \delta',\, q_0',\, \Sigma_3,\, \tau')$ generating $(S_{2n+1} \pmod 3)_{n\geq 0}$. With input $(n)_3$ we just have to simulate the previous automaton with input $(n+1)_3$, since $2(n+1)-1 = 2n+1$. The idea is to guess the digit where the ternary representations of n and $n+1$ differ. After the difference the representations of n and $n+1$ must consist only of 2's and 0's, respectively. This simulation is done by denoting the state by a pair xy, where x is the state of M_{2n-1} with input $(n)_3$ and y is the state with input $(n+1)_3$. The initial state is q_0q_1 and the transitions are defined by the rule

$$\delta'(xy), a) = \begin{cases} \delta(x,a)\delta(y,0) & \text{if } a = 2, \\ \delta(x,a)\delta(x,a+1) & \text{otherwise.} \end{cases}$$

The output is the output of the second coordinate $\tau'(xy) = \tau(y)$. We obtain the automaton presented in Figure 12, which is minimized (see Section 1.4) in Figure 13.

The corresponding 3-uniform morphism φ_1 on the alphabet $\{a, b, c, d\}$ and the coding τ_1 are

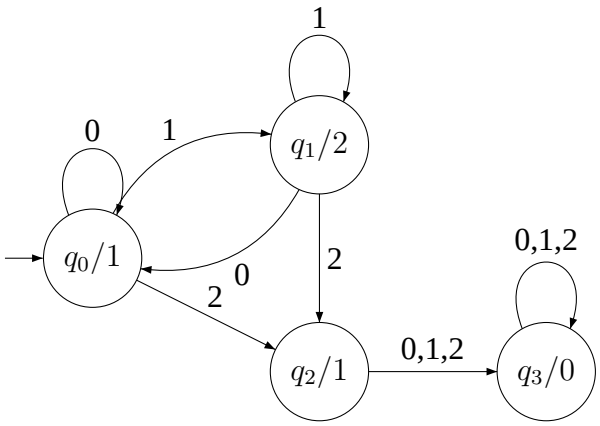


Figure 11: Automaton generating $(S_{2n-1} \pmod 3)_{n\geq 1}$

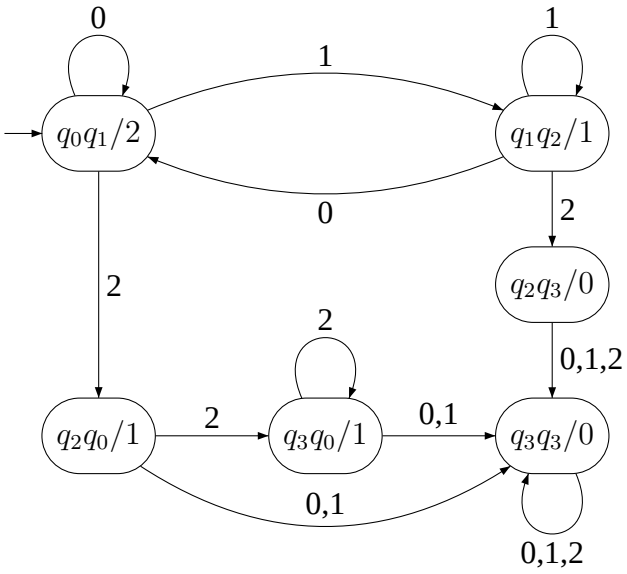


Figure 12: Automaton generating $(S_{2n+1})_{n\geq 0}$

$$\varphi_1 : \begin{array}{ccc} a & \mapsto & abc \\ b & \mapsto & abd \\ c & \mapsto & ddc \\ d & \mapsto & ddd \end{array} \qquad \tau_1 : \begin{array}{ccc} a & \mapsto & 2 \\ b & \mapsto & 1 \\ c & \mapsto & 1 \\ d & \mapsto & 0 \end{array}$$

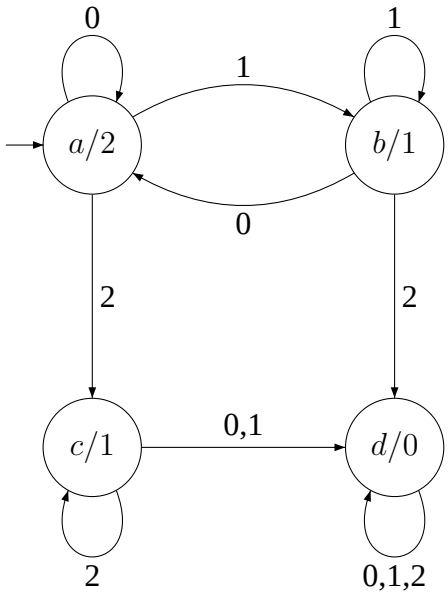


Figure 13: Minimized automaton generating $(S_{2n+1})_{n\geq 0}$

Next we make the perfect shuffle of these sequence $\mathbf{s}_{2n}$ and $\mathbf{s}_{2n+1}$ in order to obtain the sequence $\mathbf{s} = (S_n)_{n\geq 0}$. We use the algorithm PERFECT_SHUFFLE_2. Phase 1 gives the following morphisms:

$$\varphi_0' : \begin{array}{lcccc} (1,0) & \mapsto & (1,0) & (1,1) & (0,0) \\ (1,1) & \mapsto & (0,1) & (0,0) & (0,1) \\ (0,0) & \mapsto & (0,0) & (0,1) & (0,0) \\ (0,1) & \mapsto & (0,1) & (0,0) & (0,1) \end{array} \qquad \varphi_1' : \begin{array}{lcccc} (a,0) & \mapsto & (a,0) & (a,1) & (b,0) \\ (a,1) & \mapsto & (b,1) & (c,0) & (c,1) \\ (b,0) & \mapsto & (a,0) & (a,1) & (b,0) \\ (b,1) & \mapsto & (b,1) & (d,0) & (d,1) \\ (c,0) & \mapsto & (d,0) & (d,1) & (d,0) \\ (c,1) & \mapsto & (d,1) & (c,0) & (c,1) \\ (d,0) & \mapsto & (d,0) & (d,1) & (d,0) \\ (d,1) & \mapsto & (d,1) & (d,0) & (d,1) \end{array}$$

Only the images of the letters needed for the generation of the fixed point of the cartesian product $\varphi_0' \times \varphi_1'$ are presented in the following.

$$
\begin{array}{rcl}
\big((1,0),(a,0)\big) & \mapsto & \big((1,0),(a,0)\big)\big((1,1),(a,1)\big)\big((0,0),(b,0)\big) \\
\big((1,1),(a,1)\big) & \mapsto & \big((0,1),(b,1)\big)\big((0,0),(c,0)\big)\big((0,1),(c,1)\big) \\
\big((0,0),(b,0)\big) & \mapsto & \big((0,0),(a,0)\big)\big((0,1),(a,1)\big)\big((0,0),(b,0)\big) \\
\big((0,1),(b,1)\big) & \mapsto & \big((0,1),(b,1)\big)\big((0,0),(d,0)\big)\big((0,1),(d,1)\big) \\
\big((0,0),(c,0)\big) & \mapsto & \big((0,0),(d,0)\big)\big((0,1),(d,1)\big)\big((0,0),(d,0)\big) \\
\big((0,1),(c,1)\big) & \mapsto & \big((0,1),(d,1)\big)\big((0,0),(c,0)\big)\big((0,1),(c,1)\big) \\
\big((0,0),(a,0)\big) & \mapsto & \big((0,0),(a,0)\big)\big((0,1),(a,1)\big)\big((0,0),(b,0)\big) \\
\big((0,1),(a,1)\big) & \mapsto & \big((0,1),(b,1)\big)\big((0,0),(c,0)\big)\big((0,1),(c,1)\big) \\
\big((0,0),(d,0)\big) & \mapsto & \big((0,0),(d,0)\big)\big((0,1),(d,1)\big)\big((0,0),(d,0)\big) \\
\big((0,1),(d,1)\big) & \mapsto & \big((0,1),(d,1)\big)\big((0,0),(d,0)\big)\big((0,1),(d,1)\big)
\end{array}
$$

Coding τ is obtained using the rule of the algorithm. Thus simplifying the notation we have:

$$
\begin{array}{rcl@{\qquad}rcl}
\varphi: & A & \mapsto ABC & \tau: & A & \mapsto 1 \\
& B & \mapsto DEF & & B & \mapsto 2 \\
& C & \mapsto GHC & & C & \mapsto 0 \\
& D & \mapsto DIJ & & D & \mapsto 1 \\
& E & \mapsto IJI & & E & \mapsto 0 \\
& F & \mapsto JEF & & F & \mapsto 1 \\
& G & \mapsto GHC & & G & \mapsto 0 \\
& H & \mapsto DEF & & H & \mapsto 2 \\
& I & \mapsto IJI & & I & \mapsto 0 \\
& J & \mapsto JIJ & & J & \mapsto 0
\end{array}
$$

The fixed point

$$\tau(\varphi^{\omega}(A)) = 1\,2\,0\,1\,0\,1\,0\,2\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,2\,0\cdots$$

is $(S_n \pmod 3)_{n\geq 0}$. For example, $S_{17} = 111818026018 \equiv 1 \pmod 3$.

Now using Theorem 18 we can construct a 3-DFAO from the 3-uniform morphism and the coding above. Minimizing this automaton leads to the automaton represented in the Figure 7. Note that in this case the algorithm PERFECT_SHUFFLE_2 did not give the simplest morphism with respect to the number of letters.

4.5 On complexities of the algorithms

In this subsection we consider complexities of the previous algorithms. First we define what we mean by complexity. In our algorithms the input mainly consist of triplets (τ, φ, q_0), where $\tau\colon \Gamma^* \to \Sigma^*$ is a coding, $\varphi\colon \Gamma^* \to \Gamma^*$ is a k-uniform morphism and q_0 is the first symbol of the considered fixed point of φ. Consider the size of a representation of such a triplet, which clearly depends on the number of letters in the alphabet Γ. Suppose that we present the triplet (τ, φ, q_0) as a

composition of triplets $(a, \varphi(a), \tau(a))$ for each letter $a \in \Gamma$ and fix that the first letter of the first triplet is q_0. Thus the size of the representation is $|\Gamma| \times (k + 2)$ symbols. The algorithms PERFECT_SHUFFLE_1, PERFECT_SHUFFLE_2 and SUBSEQUENCE modify such representations and make cartesian products of these. The number of operations made in each stage of the algorithm is directly proportional to the size of these presentations. Thus the size of the output triplet compared to the size of the input is an appropriate measure of complexity. In other words, the *complexity function* C_X of a given algorithm X is a mapping from $\mathbb{N}$ into $\mathbb{N}$ such that $C_X(n)$ is the maximal number of letters of the output when n is the number of letters of the input.

First, we consider the algorithm PERFECT_SHUFFLE_1. Let m_i be the number of letters in Γ_i for $i = 0, 1, \ldots, k-1$. The size of the input is $n = (k+2)\sum_{i=0}^{k-1} m_i$, since all the morphisms φ are k-automatic. In state 1 of the algorithm we make new alphabets $\overline{\Gamma}_i$. Every letter of Γ_i induces one new letter. Thus we double the number of letters, i.e. $|\Gamma'_i| = 2|\Gamma_i|$. The number of letters in the output triplet (τ, φ, q_0) is $(k+2)\prod_{i=0}^{k-1}(2m_i)$. Denote the complexity function of this algorithm by C_1. We assume that k is fixed. The size of the output satisfies

$$(k+2)\prod_{i=0}^{k-1}(2m_i) \leq (k+2)\left(\frac{\sum_{i=0}^{k-1} 2m_i}{k}\right)^k = (k+2)\left(\frac{2n}{(k+2)k}\right)^k.$$

Thus we have an approximation

$$C_1(n) \leq \left(\frac{1}{k+2}\right)^{k-1}\left(\frac{2}{k}\right)^k n^k.$$

Secondly, consider the algorithm PERFECT_SHUFFLE_2. Suppose that $m_i = |\Gamma_i|$ for $i = 0, 1, \ldots, m-1$ like above. The size of the input is now $(k+2)\sum_{i=0}^{m-1} m_i$. In the algorithm we define $\Gamma'_i = \Gamma_i \times \Sigma_m$. Hence $|\Gamma'_i| = m \cdot m_i$ and the size of the output is $(k+2)\prod_{i=0}^{m-1}(m \cdot m_i)$. Denote the complexity function of PERFECT_SHUFFLE_2 by C_2. Using similar considerations as above, we can prove that

$$C_2(n) \leq \left(\frac{1}{k+2}\right)^{m-1} n^m.$$

If we compare the two algorithms in the case, where $m = k$, we note that PERFECT_SHUFFLE_1 has lower complexity than PERFECT_SHUFFLE_2 if $k > 2$. In the case $k = 2$ they are equal.

Finally, we analyze the complexity of the algorithm SUBSEQUENCE. Let the size of Γ be m. Then the input consists of the representation of the triplet (τ, φ, q_0) with $(k+2)m$ symbols and $2(l+1)$ symbols representing the numbers a and b. Thus the size of Γ_{ab} is m^2k^{l+2} and the size of the output triplet $(\tau_{ab}, \varphi_{ab}, q_{0ab})$ is $(k+2)m^2k^{l+2}$. Suppose that l is fixed and denote the complexity function of this algorithm by C_3. We define

$$C_3(n) = \max\{(k+2)m^2k^{l+2}\},$$

where the maximum is taken over all $m \leq \frac{n-2(l+1)}{k+2}$. Hence we may say that all the three algorithms have polynomial complexity.

4.6 Final Remarks

Let us consider shuffles of automatic sequences from a couple of other viewpoints. To begin with, we study the following problem related to fixed points of morphisms:

Problem 1. Suppose that $\mathbf{w} = f_2(f_1^\omega(a))$, where $f_1 \colon \Sigma^* \to \Sigma^*$ is a k-uniform and $f_2 \colon \Sigma^* \to \Delta^*$ is an l-uniform morphism. Represent $\mathbf{w}$ in the form $f_2'(f_1'^\omega(a))$, where f_1' is still a k-uniform morphism but f_2' is a coding.

It follows from Corollary 3 that such a presentation always exists. Finding the representation is actually equivalent to the question of calculating the generating triplet of the perfect l-shuffle of k-automatic sequences. Namely, let $\mathbf{v} = v_0v_1v_2\cdots = f_1^\omega(a)$. Then $\mathbf{w} = f_2(\mathbf{v}) = f_2(v_0)f_2(v_1)f_2(v_2)\cdots$. Define now codings τ_i, $0 \leq i \leq l-1$ by the rule $\tau_i(a) = f_2(a)(i)$ for all $a \in \Sigma$. By this definition, we have

$$f_2(\mathbf{v}) = \tau_0(v_0)\tau_1(v_0)\cdots\tau_{l-1}(v_0)\tau_0(v_1)\tau_1(v_1)\cdots\tau_{l-1}(v_1)\cdots$$

This is the perfect l-shuffle of k-automatic sequences $\tau_i(f_1^\omega(a))$, $0 \leq i \leq l-1$ and our algorithm PERFECT_SHUFFLE_2 with input (τ_i, f_1, a), $0 \leq i \leq l-1$ gives the desired triplet (f_2', f_1', a).

Secondly, we make a generalization to our notion of regular shuffles. Suppose that the shuffle is made according to a special infinite rule called a *directive sequence*. This means that we read letter by letter this directive sequence and all the sequences to be shuffled. The directive sequence tells us from which sequence the next letter of the shuffle comes from. More formally, let us shuffle sequences $\mathbf{u}^{(i)} = (u_n^{(i)})_{n\geq 0}$, $0 \leq i \leq m-1$ according to the directive sequence $\alpha = r_0r_1r_2\cdots$ over the alphabet Σ_m. Denote $\alpha_i = r_0r_1\cdots r_{i-1}$ for $i \geq 1$ and let $\alpha_0 = \varepsilon$. The resulting shuffle $\mathbf{u} = u_0u_1u_2\cdots$ satisfies the rule

$$u_i = u^{(r_i)}_{|\alpha_i|_{r_i}}.$$

and it is called the *shuffle of the sequences* $\mathbf{u}^{(i)}$ *with directive sequence* α . For example, suppose that $m = 3$ and $\alpha = 00122201\cdots$. Then the shuffle of $\mathbf{u}^{(i)}$, $0 \leq i \leq 2$ with directives sequence α begins with

$$u_0^{(0)}u_1^{(0)}u_0^{(1)}u_0^{(2)}u_1^{(2)}u_2^{(2)}u_2^{(0)}u_1^{(1)}$$

Compare the previous definition to the definition of regular shuffles. The rule of a regular shuffle, which is a finite word, is now replaced by an infinite word. In fact, regular shuffles are just shuffles with periodic directive sequence. When considering automatic sequences we have the following closure property, which follows directly from Theorem 41.

Theorem 45. *The shuffle of automatic sequences with periodic directive sequence is an automatic sequence.*

In the theorem we may even replace the periodic directive sequence with an ultimately periodic sequence since the class of automatic sequences is closed under shifts (Theorem 25). This means that after a finite prefix we may construct the shuffle according to a periodic directive sequence using suffices of the original sequences, i.e. shifted automatic sequences. Now we could try to generalize this result and consider the following question.

Problem 2. Is the shuffle of automatic sequences with automatic directive sequence always automatic?

The answer is unfortunately negative. We construct the following counter example.

Example 8. Suppose that $\mathbf{u}^{(0)}$ is a sequence of zeros and $\mathbf{u}^{(1)}$ is a characteristic sequence of powers of 2.

$$u_n^{(1)} = \begin{cases} 1 & \text{if } n = 2^k \text{ for } k \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Thus we have

$$\mathbf{u}^{(1)} = 0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\cdots$$

Both these sequences $\mathbf{u}^{(0)}$ and $\mathbf{u}^{(1)}$ are clearly 2-automatic. Suppose now that the directive sequence α is equal to $\mathbf{u}^{(1)}$. Because 1 occurs only in the sequence $\mathbf{u}^{(1)}$ in the position 2^k and letters from sequence $\mathbf{u}^{(1)}$ are only in every 2^kth position of the shuffle $\mathbf{u} = u_0u_1u_2\cdots$, we have

$$u_n = \begin{cases} 1 & \text{if } n = 2^{2^k} \text{ for } k \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

But this sequence is not automatic, which can be easily proved by pumping lemma. Otherwise there would exist an automaton recognizing the language $L = \{10^{2^j} \mid j \geq 0\}$. By pumping lemma, there exists a word of the form uvw such that $v \neq \varepsilon$ and uv^iw belongs to L for every $i \geq 0$. Now v cannot contain 1, since words of the language contain only one letter 1. Thus v is of the form 0^s. Suppose that $u = 10^r$, where $r \geq 0$. Now $uv^iw = 10^r0^{si}0^{2^j-(r+s)} = 10^{2^j+s(i-1)} \in L$ for every $i \geq 0$. This means that all numbers $2^j + s(i-1)$ should be powers of two, which is clearly impossible.

References

[1] B. Adamczewski, Y. Bugeaud, and F. Luca, Sur la complexité des nombres algébriques. C. R. Acad. Sci. Paris, ser. I **339**, 11–14, 2004.

[2] J.-P. Allouche, Automates finis en théorie des nombres. Exposition. Math. **5**, 239–266, 1987.

[3] J.-P. Allouche and J. Shallit, The ubiquitous Prouhet-Thue-Morse sequence. In C. Ding, T. Helleseth, and H. Niederreiter, editors, *Sequences and Their Applications, Proceedings of SETA '98*, 1-16, Springer-Verlag, 1999.

[4] J.-P. Allouche and J. Shallit, Automatic Sequences: Theory, Applications, Generalizations. Cambridge University Press, 387–391, 2003.

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

[6] J. A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events. In *Mathematical Theory of Automata, Volume 12 of MRI Symposia Series*, Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 529–561, 1962.

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

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

[9] F. Durand, A characterization of substitutive sequences using return words. Discrete Math. **179**, no. 1-3, 89–101, 1998.

[10] S. Eilenberg, Automata, Languages, and Machines. Vol. A. Academic Press, 1974.

[11] S. Ferenczi and C. Mauduit, Transcendence of numbers with a low complexity expansion. J. Number Theory **67**, 146–161, 1997.

[12] G. Hansel, A propos d'un théorème de Cobham. In D. Perrin, editor, *Actes de la Fête des Mots, Greco de Programmation*, CNRS, Rouen, 55–59, 1982

[13] G.A. Hedlund and M. Morse, Symbolic dynamics II: Sturmian trajectories. Amer. J. Math. **62** 1–42, 1940.

[14] J.E. Hopcroft, An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations, Academic Press*, 189–196, 1971

[15] T. Kärki, Transcendence of numbers with an expansion in a subclass of complexity $2n + 1$. TUCS Tech. Rep. 654, Turku Centre for Computer Science, Finland, 1–13, December 2004.

[16] T. Kärki, A note on the proof of Cobham's theorem. TUCS Tech. Rep. 713, Turku Centre for Computer Science, Finland, 1–4, September 2005.

[17] M. Lothaire, Applied Combinatorics on Words. Cambridge University Press, 26–35, 2005.

[18] C. Michaux and R. Villemaire, Cobham's theorem seen through Büchi's theorem. In *Proc. 20th Int. Conf. on Automata, Languages, and Programming (ICALP), Vol. 700 of Lecture Notes in Computer Science*, 325–334, Springer-Verlag, 1993.

[19] C. Michaux and R. Villemaire, Presburger arithmetic and recognizability of sets of natural numbers by automata: New proofs of Cobham's and Semenov's theorems. Ann. Pure Appl. Logic **77**, 251–277, 1996.

[20] M. Minsky and S. Papert, Unrecognizable sets of numbers. J. Assoc. Comput. Mach. **13**, 281–286, 1966.

[21] F.R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata. IEEE Trans. Comput. **20**, 1211–1214, 1971.

[22] A. Muchnik, Definable criterion for definability in Presburger Arithmetic and its application. *preprint in Russian*, Institute of New Technologies, 1991.

[23] N.J.A. Sloane, The On-Line Encyclopedia of Integer Sequences http://www.research.att.com/~njas/sequences/index.html

[24] E. Prouhet, Mémoire sur quelques relations entre les puissances des nombres. C. R. Acad. Sci. Paris, ser. I **33**, 225, 1851.

[25] C. Reutenauer, Démonstration du théorème de Cobham sur les ensembles de nombres reconnaissables par automate fini, d'après Hansel. In *Séminaire d'Informatique Théorique, Année 1983-84, Université Paris 7, Paris*, 217–224, 1984.

[26] M. Rigo and L. Waxweiler, A note on syndeticity, recognizable sets and Cobham's theorem. To appear in Bull. EATCS.

[27] R.M. Ritchie, Finite automata and the set of squares. J. Assoc. Comput. Mach. **10**, 528-531, 1963.

[28] A.L. Semenov, Presburgerness of predicates regular in two number systems (in Russian). Sibirsk. Mat. Zh. **18**, 403–418, 1977, English translation, Siberian Math. J. **18**, 289-299, 1997.

[29] J. Shallit, Numeration systems, linear recurrences, and regular sets. Inform. and Comput. **113**, no. 2, 331–347, 1994.

[30] L. Shapiro, Elementary problem E 3343. Amer. Math. Monthly **96**, 734, 1989. Solution by Carl Schoen and Paolo Ranaldi, Amer. Math. Monthly **98**, 368, 1991.

[31] M. Quéffelec, Une nouvelle propriété des suites de Rudin-Shapiro. Annales de l'institut Fourier **37** no. 2, 115–138, 1987.

[32] B. Watson, Taxonomies and Toolkits of Regular Language Algorithms. Ph.D. thesis, Department for Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1995.

[33] B.M.M. de Weger, Elementary problem E 3470. Amer. Math. Monthly **98**, 955, 1991. Solution by M. Vowe and O.P. Lossers, Amer. Math. Monthly **101**, 83-84, 1994. Also see **102**, 936, 1995.

Index

alphabet, 3
automatic numbers, 34
automatic sequence, 19

block compression, 31
block substitution, 31

cartesian product, 50
catenation, 3
characteristic function, 3
characteristic sequence, 3
coding, 4
complexity function, 69

deterministic finite automaton, 4
deterministic finite automaton with output, 10
DFA, 4
DFAO, 10
directive sequence, 70

factor, 3
fiber, 10
finite-state function, 10
fixed point, 25

intermediate sum, 59

k-fiber, 22
kernel, 23

language, 3
length, 3
letter, 3

minimal automaton, 12
morphic sequence, 25
morphism, 4
multiplication rows, 58
multiplicatively dependent, 35
multiplicatively independent, 35
Myhill-Nerode equivalence, 12

NFA, 5
nondeterministic finite automaton, 5
nonerasing, 4
normalized representation, 7

output function, 10

partial products, 58
perfect m-shuffle, 50
periodic deletion, 30
prefix, 3
prolongable, 25
properly recurrent, 40
pure morphic sequence, 25

rational expression, 6
rational language, 6
recognizable language, 4
recurrence, 32
recurrent state, 40
regular language, 4
regular shuffle, 49
reversal, 3
right congruence, 12
right dense, 38
right-invariant, 12
Rudin-Shapiro sequence, 21

Schröder numbers, 21
shift, 30
stable, 42
subset construction, 6
subword complexity, 31
suffix, 3
syndetic, 38

Thue-Morse sequence, 19
transition function, 4
transition relation, 5

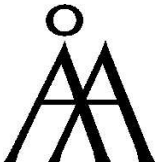
ultimately periodic, 3
uniform morphism, 4
uniformly morphic sequence, 25

word, 3

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1688-3
ISSN 1239-1891