Automatic Theorem-Proving in Combinatorics on Words

Daniel Goč, Dane Henshall, and Jeffrey Shallit

School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1 Canada
{dhenshall,dgoc,shallit}@uwaterloo.ca

Abstract. We describe a technique for mechanically proving certain kinds of theorems in combinatorics on words, using finite automata and a package for manipulating them. We illustrate our technique by applying it to (a) solve an open problem of Currie and Saari on the lengths of unbordered factors in the Thue-Morse sequence; (b) verify an old result of Prodinger and Urbanek on the paperfolding sequence and (c) find an explicit expression for the recurrence function for the Rudin-Shapiro sequence. All results were obtained by machine computations.

Dedicated to the memory of Sheng Yu (1950–2012): friend and colleague

1 Introduction

The title of this paper is a bit of a pun. On the one hand, we are concerned with certain natural questions about *automatic sequences*: sequences over a finite alphabet where the n'th term of the sequence is expressible as a finite-state function of the base-k representation of n. On the other hand, we are interested in answering these questions purely mechanically, in an *automated* fashion.

Let $\mathbf{x} = (a(n))_{n\geq 0}$ be an infinite sequence over a finite alphabet Δ. Then $\mathbf{x}$ is said to be k-*automatic* if there is a deterministic finite automaton M taking as input the base-k representation of n, and having $a(n)$ as the output associated with the last state encountered [2]. In this case, we say that M *generates* the sequence $\mathbf{x}$.

We write $\mathbf{x}[i] = a(i)$, and we let $\mathbf{x}[i..i+n-1]$ denote the *factor* of length n beginning at position i in $\mathbf{x}$. A sequence is said to be *squarefree* if it contains no factor of the form xx, where x is a nonempty word, and is said to *overlapfree* if it contains no factor of the form $ayaya$, where a is a single letter and y is a possibly empty word.

In Figure 1, we give, as an example, an automaton generating the well-known Thue-Morse sequence $\mathbf{t} = t(0)t(1)t(2)\cdots = \mathtt{011010011001}\cdots$ [3]. The input is n, expressed in base 2, and the output is the number contained in the state last reached. Thus $t(n)$ is the sum, modulo 2, of the binary digits of n. In a celebrated result, Thue proved [27,28,4] that the sequence $\mathbf{t}$ is overlapfree.

For at least 25 years, researchers have been interested in the algorithmic decidability of assertions about automatic sequences. For example, in one of the earliest results, Honkala [19] showed that, given an automaton M, it is decidable if the sequence generated by M is ultimately periodic.

N. Moreira and R. Reis (Eds.): CIAA 2012, LNCS 7381, pp. 180–191, 2012.

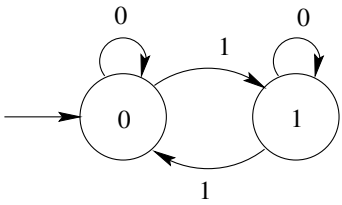

Fig. 1. A finite automaton generating the Thue-Morse sequence

Recently, Allouche et al. [1] found a different proof of Honkala's result using a more general technique. Using this technique, they were able to give algorithmic solutions to many classical problems from combinatorics on words such as

Given an automaton, is the generated sequence squarefree? Or overlapfree?

The technique of Allouche et al. is at its core, very similar to work of Büchi, Bruyère, Michaux, Villemaire, and others, involving formal logic; see, e.g., [5]. The basic idea is as follows: given the automaton M, and some predicate $P(n)$ we want to check, we alter M by a series of transformations to a new automaton M' that accepts the base-k representations of those integers n for which $P(n)$ is true. Then we can check the assertion "$\exists\, n\ P(n)$" simply by checking if M' accepts anything (which can be done by a standard depth-first search on the underlying directed graph of the automaton). We can check the assertion "$\forall\, n\ P(n)$" by checking if M' accepts everything. And we can check assertions like "$P(n)$ holds for infinitely many n" by checking if M' has a reachable cycle from which a final state is reachable.

Using this idea, Allouche et al. were able to show to reprove, purely mechanically using a computer program, Thue's classic result on the overlapfreeness of the the Thue-Morse sequence.

Later, the technique was applied to give decision procedures for other properties of automatic sequences. For example, Charlier et al. [6] showed that it can be used to decide if a given k-automatic sequence

- contains powers of arbitrarily large exponent;
- is recurrent;
- is uniformly recurrent.

A sequence is said to be *recurrent* if every factor that occurs, occurs infinitely often. A sequence $\mathbf{x}$ is said to be *uniformly recurrent* if it is recurrent and furthermore for each finite factor w occurring in $\mathbf{x}$, there is a constant $c(w)$ such that two consecutive occurrences of w are separated by at most $c(w)$ positions.

More recently, variations of the technique have been used to

- compute the critical exponent;
- compute the initial critical exponent;
- decide if a sequence is linearly recurrent;
- compute the Diophantine exponent.

(For definitions of these terms see [25].)

2 The Decision Procedure

In [6] we have the following theorem:

Theorem 1. *If we can express a property of a k-automatic sequence* $\mathbf{x}$ *using quantifiers, logical operations, integer variables, the operations of addition, subtraction, indexing into* $\mathbf{x}$*, and comparison of integers or elements of* $\mathbf{x}$*, then this property is algorithmically decidable.*

Let us outline how the decision procedure works. First, the input to the decision procedure: an automaton $M = (Q, \Sigma_k, \Delta, \delta, q_0, \tau)$ generating the k-automatic sequence $\mathbf{x}$. Here

- Q is a nonempty set of states;
- $\Sigma_k := \{0, 1, \ldots, k-1\}$;
- Δ is the output alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function;
- q_0 is the initial state; and
- $\tau : Q \to \Delta$ is the output mapping.

In this paper, we assume that the automaton takes as input the representation of n in base k, *starting with the least significant digit*; we call this the *reversed representation* of n and write it as $(n)_k$. We allow leading zeroes in the representation (which, because of our convention, are actually trailing zeroes). Thus, for example, 011 and 01100 are both acceptable representations for 6 in base 2.

We might also need to encode pairs, triples, or r-tuples of integers. We handle these by first padding the reversed representation of the smaller integer with trailing zeroes, and then coding the r-tuple as a word over Σ_k^r. For example, the pair $(20, 13)$ could be represented in base-2 as

$$[0, 1][0, 0][1, 1][0, 1][1, 0],$$

where the first components spell out 00101 and the second components spell out 10110. Of course, there are other possible representations, such as

$$[0, 1][0, 0][1, 1][0, 1][1, 0][0, 0],$$

which correspond to non-canonical representations having trailing zeroes; these are also permitted.

Rather than present a detailed proof, we illustrate the idea of the decision procedure in the proof of the following new result:

Theorem 2. *The following problem is algorithmically decidable: given two k-automatic sequences* $\mathbf{x}$ *and* $\mathbf{y}$*, generated by automata M_1 and M_2, respectively, decide if* $\mathbf{x}$ *is a shift of* $\mathbf{y}$ *(that is, decide if there exists a constant c such that $\mathbf{x}[n] = \mathbf{y}[n + c]$ for all $n \geq 0$).*

Proof. We first create an NFA M that accepts the language

$$\{(c)_k \ : \ \exists \ n \text{ such that } \mathbf{x}[n] \neq \mathbf{y}[n + c]\}.$$

To do so, on input $(c)_k$, M

- guesses $w_1 = (n)_k$ nondeterministically (perhaps with trailing zeroes appended),
- simulates M_1 on w_1,
- adds n to c and computes the base-k representation of $w_2 = (n + c)_k$ digit-by-digit "on the fly", keeping track of carries, as necessary, and simulates M_2 on w_2, and
- accepts if the outputs of both machine differ.

We now convert M to a DFA M', and change final states to non-final (and vice versa). Then M' accepts the language

$$\{(c)_k \ : \ \mathbf{x}[n] = \mathbf{y}[n + c] \text{ for all } n \geq 0\}.$$

Thus, $\mathbf{x}$ is a shift of $\mathbf{y}$ if and only if M' accepts any word, which is easily checked through depth-first search. □

Remark 1. As we can see, the size of the automata involved depends, in an unpleasant way, on the number of quantifiers needed to state the logical expression characterizing the property being checked, because existential quantifiers are implemented through nondeterminism, and universal quantifiers are implemented through nondeterminism and complementation (which is implemented in a DFA by exchange of the role final and non-final states). Thus each new quantifier could increase the current number of states, say n, to 2^n using the subset construction. If the original automata have at most N states, it follows that the running time is bounded by an expression of the form

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{p(N)}}}}}}$$

where p is a polynomial and the number of exponents in the tower is one less than the number of quantifiers in the logical formula characterizing the property being checked.

This extraordinary computational complexity raises the natural question of whether the decision procedure could actually be implemented for anything but toy examples. Luckily the answer seems to be yes — at least in some cases — as we will see below.

The algorithms we discuss were implemented by the first two authors, independently, using two different programs. The results in Sections 3 and 4 have been double-checked with these separate implementations, which should give some confidence about the results.

Remark 2. Prior art: as a referee points out, very similar ideas are contained in the work of Glenn and Gasarch [14,15] on implementing a decision procedure for WS1S, the weak second-order theory of one successor. The main differences between their work and ours are (a) we work with base-k encodings of integers, instead of unary encodings, and (b) we apply our ideas to solve some interesting open problems about automatic sequences, instead of checking randomly-generated sentences.

3 Borders

A word w is *bordered* if it begins and ends with the same word x with $0 < |x| \leq |w|/2$; Otherwise it is *unbordered*. An example in English of a bordered word is `entanglement`. A bordered word is also called *bifix* in the literature, and unbordered words are also called *bifix-free* or *primary*.

Bordered and unbordered words have been actively studied in the literature, particularly with regard to the Ehrenfeucht-Silberger problem; see, for example, [13,20,10,11,16,17,7,18,22,12], just to name a few.

Currie and Saari [8] studied the unbordered factors of the Thue-Morse sequence $\mathbf{t}$. They proved that if $n \not\equiv 1 \pmod 6$, then $\mathbf{t}$ has an unbordered factor of length n. (Also see [24, Lemma 4.10 and Problem 4.1].) However, this is not a necessary condition, as

$$\mathbf{t}[39..69] = \texttt{00110100101101001100101101001 01},$$

which is an unbordered factor of length 31. Currie and Saari left it as an open problem to give a complete characterization of the integers n for which $\mathbf{t}$ has an unbordered factor of length n.

The following theorem and proof, quoted practically verbatim from [6], shows that, more generally, the characteristic sequence of n for which a given k-automatic sequence has an unbordered factor of length n, is itself k-automatic:

Theorem 3. *Let* $\mathbf{x} = a(0)a(1)a(2)\cdots$ *be a* k*-automatic sequence. Then the associated infinite sequence* $\mathbf{b} = b(0)b(1)b(2)\cdots$ *defined by*

$$b(n) = \begin{cases} 1, & \text{if } \mathbf{x} \text{ has an unbordered factor of length } n; \\ 0, & \text{otherwise;} \end{cases}$$

is k*-automatic.*

Proof. The sequence $\mathbf{x}$ has an unbordered factor of length n

iff

$\exists\, j \geq 0$ such that the factor of length n beginning at position j of $\mathbf{x}$ is unbordered

iff

there exists an integer $j \geq 0$ such that for all possible lengths l with $1 \leq l \leq n/2$, there is an integer i with $0 \leq i < l$ such that the supposed border of length l beginning and ending the factor of length n beginning at position j of $\mathbf{x}$ actually differs in the i'th position

iff

there exists an integer $j \geq 0$ such that for all integers l with $1 \leq l \leq n/2$ there exists an integer i with $0 \leq i < l$ such that $\mathbf{x}[j+i] \neq \mathbf{x}[j+n-l+i]$.

Now assume $\mathbf{x}$ is a k-automatic sequence, generated by some finite automaton. We show how to implement the characterization given above with an automaton.

We first create an NFA that given the $(j, l, n)_k$ guesses the base-k representation of i, digit-by-digit, checks that $i < l$, computes $j + i$ and $j + n - l + i$ on the

fly, and checks that $\mathbf{x}[j+i] \neq \mathbf{x}[j+n-l+i]$. If such an i is found, it accepts. We then convert this to a DFA, and interchange accepting and nonaccepting states. This DFA M_1 accepts $(j,l,n)_k$ such that there is no i, $0 \leq i < l$ such that $\mathbf{x}[j+i] = \mathbf{x}[j+n-l+i]$. We then use M_1 as a subroutine to build an NFA M_2 that on input $(j,n)_k$ guesses l, checks that $1 \leq l \leq n/2$, and calls M_1 on the result. We convert this to a DFA and interchange accepting and nonaccepting states to get M_3. Finally, this M_3 is used as a subroutine to build an NFA M_4 that on input n guesses j and calls M_3.

The characteristic sequence of these integers n is therefore k-automatic. □

Since the proof is constructive, one can, in principle, carry out the construction to get an explicit description of the lengths for which the Thue-Morse sequence has an unbordered factor.

Doing so results in the following theorem:

Theorem 4. *There is an unbordered factor of length n in $\mathbf{t}$ if and only if the base-2 representation of n (starting with the most significant digit) is not of the form $1(01^*0)^*10^*1$.*

Proof. The proof of this theorem is purely mechanical, and it involves performing a sequence of operations on finite automata. The second author wrote a program in C++, using his own automata package, to perform these operations. There are four stages to the computation, which are described in detail below.

Stage 1

Let T be the automaton of Figure 1 generating the Thue-Morse sequence $\mathbf{t}$. Stage 1 takes T as input and outputs an automaton M_1, where M_1 accepts $w \in (\{0,1\}^4)^*$ if and only if w is the base-2 representation of some $(n,j,l,i) \in S_1$, where

$$S_1 = \{(n,j,l,i) \ : \ 0 < l \leq n/2 \text{ and } i < j \text{ and } \mathbf{t}[j+i] \neq \mathbf{t}[n+j-l+i]\}. \quad (1)$$

The size of M_1 was only 102 states. However, since the input alphabet for M_1 is of size $2^4 = 16$, a considerable amount of complexity is being stored in the transition matrix. Stage 1 passed all 1.3 million tests meant to ensure that M_1 corresponds to S_1.

Stage 2

The purpose of Stage 2 is to remove the variable i by simulating it. The resulting machine, after being negated, accepts (n,j,l) iff the length n factor of t starting at index j has a border of length l. So Stage 2 produces the automaton M_2, which is the negation of the result of simulating i. More formally, M_2 accepts a word $w \in (\{0,1\}^3)^*$ if and only if w is the base-2 representation of some $(n,j,l) \in S_2$, where

$$S_2 = \{(n,j,l) : \nexists \ i \text{ for which } (n,j,l,i) \in S_1\} \quad (2)$$

The size of M_2 after subset construction was 8689 states, and it minimized down to 127 states. The output of Stage 2 passed all 1.6 million tests meant to ensure that M_2 corresponds to S_2.

Stage 3

The purpose of Stage 3 is to remove l by simulating it. By the end of Stage 3, most of the work has already been done. The output of Stage 3, M_3, accepts an input word $w \in (\{0,1\}^2)^*$ if and only if w is the base-2 representation of some $(n, j) \in S_3$, where

$$S_3 = \{(n, j) : \nexists\ l \text{ such that } (n, j, l) \in S_2\} \tag{3}$$

or, in other words

$$S_3 = \{(n, j) :\ \mathbf{t} \text{ has an unbordered factor of length } n \text{ at index } j\}. \tag{4}$$

The size of M_3 after subset construction was 1987 states, and it minimized down to 263 states. The output of Stage 3 passed all 1.9 million tests meant to ensure that M_3 corresponds to S_3.

Stage 4

Finally, Stage 4 simulates j on M_3 and negates the result. So the output of Stage 3 is an automaton that accepts the binary representation of a positive integer $n > 1$ if and only if the Thue-Morse word has no unbordered factor of length n. Formally put, the automaton M_4 produced by Stage 4 accepts a word $w \in \{0,1\}^*$ if and only if w is the base-2 representation of some $n \in S_4$, where

$$S_4 = \{n \in \mathbb{N} \ :\ n > 1, \nexists\ j \text{ for which } (n, j) \in S_3\}. \tag{5}$$

The size of M_4 after subset construction is 2734 states, and it minimized to 7 states. M_4 accepts the reverse of $1(01^*0)^*10^*1$. Therefore the Thue-Morse word has an unbordered factor of length n if and only if the base-2 representation of n (starting with the most significant digit) is not of the form $1(01^*0)^*10^*1$.

The total computation took 9 seconds of CPU time on a 2.9GHz Dell XPS laptop. □

Remark 3. Here are some additional implementation details.
In order to implement the needed operations on automata, we must decide on an encoding of elements of $(\Sigma_k^n)^*$. We could do this by performing a perfect shuffle of each individual word over Σ_k^*, or by letting the alphabet itself be represented by k-tuples. The decision represents a tradeoff between state size and alphabet size. We used the latter representation, since (a) it makes the algorithms considerably easier to implement and understand and (b) decreases the number of states needed.

It was mentioned earlier how many tests were passed in each stage. In order to make sure that the final automaton is what we expect, a number of tests are run after each stage on the output of that stage.

For example, let $\mathbf{x}$ be an automatic sequence. The testing framework requires a C++ function which given n computes $\mathbf{x}[n]$. Before any operations are done, the automaton given for $\mathbf{x}$ is tested against the C++ function to make sure that they match for the first 10,000 elements. Then, at each stage before Stage 4 the resulting automaton is tested to give confidence that the operations on the automata are giving the desired results.

For example, after Stage 2 of computing the set of lengths for which there exists an unbordered factor of an automatic sequence $\mathbf{x}$, we expect the machine M_2 to accept the language S_2, where

$$S_2 = \{(n, j, l) : \nexists\ i \text{ for which } \mathbf{x}[j+i] = \mathbf{x}[n+j-l+i]\} \tag{6}$$

This is then tested by making sure M_2 accepts $(n, j, l)_k$ if and only if $(n, j, l) \in S_2$ for all $n, j, l \leq 1400$. These tests were invaluable to debugging, and provide confidence in the final result of the computation.

Finally, we have to address the issue of multiple representations. It is easy to forget that automata accept words in ${\Sigma_k}^*$, and not integers. For some operations, such as complement and intersection, it is crucial that if one binary representation is accepted by the automaton, then all binary representations must be accepted.

4 Additional Results on Unbordered Words

We also applied our decision procedure above to two other famous sequences: the Rudin-Shapiro sequence [23,26] and the paperfolding sequence [9].

For a word $w \in 1(0+1)^*$, we define $a_w(n)$ to be the number of (possibly overlapping) occurrences of w in the (ordinary, unreversed) base-2 representation of n. Thus, for example, $a_{11}(7) = 2$.

The *Rudin-Shapiro* sequence $\mathbf{r} = r(0)r(1)r(2)\cdots$ is then defined to be $r(n) = (-1)^{a_{11}(n)}$. It is a 2-automatic sequence generated by an automaton of four states.

The *paperfolding sequence* $\mathbf{p} = p(0)p(1)p(2)\cdots$ is defined as follows: writing $(n)_2 00$ as $1^i 0aw$ for some $i \geq 0$ some $a \in \{0, 1\}$, and some $w \in \{0, 1\}^*$, we have $p(n) = (-1)^a$. It is a 2-automatic sequence generated by an automaton of four states.

Theorem 5. *The Rudin-Shapiro sequence has an unbordered factor of every length.*

Proof. We applied the same technique discussed previously for the Thue-Morse sequence.

Here is a summary of the computation:

Stage 1: 269 states
Stage 2: 85313 states minimized to 1974
Stage 3: 48488 states minimized to 6465
Stage 4: 6234 states.

The Stage 4 NFA has 6234 states. We were unable to determinize this automaton directly (using two different programs) due to an explosion in the number of states created. Instead, we reversed the NFA (creating an NFA for L^R) and determinized this instead. The resulting DFA has 30 states, and upon minimization, gives a 1-state automaton accepting all strings. □

Theorem 6. *The paperfolding sequence has an unbordered factor of length n if and only if the reversed representation $(n)_2$ is rejected by the automaton given in Figure 4.*

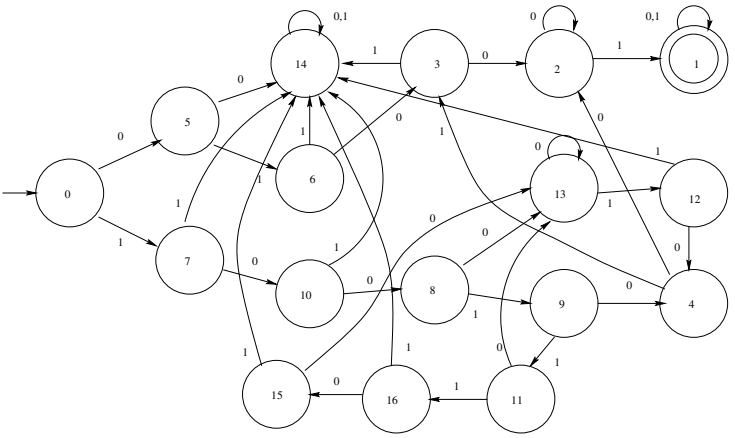

Fig. 2. A finite automaton for unbordered factors in the paperfolding word

Proof. We applied the same technique discussed previously for the Thue-Morse sequence.

Here is a summary of the computation: 6 seconds cpu time on a 2.9GHz Dell XPS laptop.

Stage 1, 159 states
Stage 2, 1751 minimized down to 89 states
Stage 3, 178 minimized down to 75 states
Stage 4, 132 minimize down to 17 states . □

5 Other Problems

We applied our technique to some other problems. First, we considered the squares in the paperfolding sequence. In 1979, Prodinger and Urbanek [21] characterized the squares in the paperfolding sequence, using a case analysis. We verified this by creating an automaton to accept the language

$$\{(n)_2 \ : \ \exists \ i \ \mathbf{p}[i..i+n-1] = \mathbf{p}[i+n..i+2n-1]\}.$$

The resulting automaton (most significant-digit first) is depicted below, from which we recover the Prodinger-Urbanek result that the only squares xx in $\mathbf{x}$ have lengths $|x| = 1, 3$, or 5.

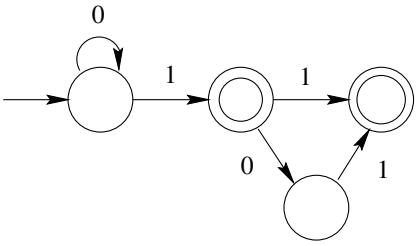

Fig. 3. Lengths of squares in the paperfolding sequence

Next, we computed a new explicit expression for the recurrence function $R_{\mathbf{r}}(n)$ and recurrence quotient for the Rudin-Shapiro sequence $\mathbf{r}$. Here $R_{\mathbf{r}}(n)$ is the smallest integer m such that every factor of $\mathbf{r}$ of length m contains as a factor all the factors of length n. Allouche and Bousquet-Mélou gave the estimate $R_{\mathbf{r}}(n+1) < 172n$ for $n \geq 1$. (Actually, their result was more general, as it applies to any "generalized" Rudin-Shapiro sequence.) We used our method to prove the following result:

Theorem 7. *Let* $\mathbf{r} = (r(n))_{n\geq 0}$ *be the Rudin-Shapiro sequence. Then*

$$R_{\mathbf{r}}(n) = \begin{cases} 5, & \text{if } n = 1; \\ 19, & \text{if } n = 2; \\ 25, & \text{if } n = 3; \\ 20 \cdot 2^t + n - 1, & \text{if } n \geq 4 \text{ and } t = \lceil \log_2(n-1) \rceil. \end{cases}$$

Furthermore, the recurrence quotient

$$\sup_{n\geq 1} \frac{R_{\mathbf{r}}(n)}{n}$$

is equal to 41*; it is not attained.*

Proof. We created a DFA to accept

$$\{(m,n)_2 \ : \ (m - 20\cdot 2^t - n + 1, n) \ : \ n \geq 4 \text{ and } m = R(n) \text{ and } t = \lceil \log_2(n-1) \rceil\}.$$

We then verified that the resulting DFA accepted only pairs of the form $(0,n)_2$ for $n \geq 4$.

For the recurrence quotient, the local maximum is evidently achieved when $n = 2^r + 2$ for some $r \geq 1$; here it is equal to $(41 \cdot 2^r + 2)/(2^r + 2)$. As $r \to \infty$, this clearly approaches 41 from below. □

6 Open Problems

Which of the problems mentioned in § 1 are algorithmically decidable for the more general class of morphic sequences?

Can the techniques be applied to detect abelian powers in automatic sequences?

References

1. Allouche, J.P., Rampersad, N., Shallit, J.: Periodicity, repetitions, and orbits of an automatic sequence. Theoret. Comput. Sci. 410, 2795–2803 (2009)
2. Allouche, J.P., Shallit, J.: Automatic Sequences: Theory, Applications, Generalizations. Cambridge University Press (2003)
3. Allouche, J.P., Shallit, J.O.: The ubiquitous Prouhet-Thue-Morse sequence. In: Ding, C., Helleseth, T., Niederreiter, H. (eds.) Sequences and Their Applications, Proceedings of SETA 1998, pp. 1–16. Springer, Heidelberg (1999)
4. Berstel, J.: Axel Thue's Papers on Repetitions in Words: a Translation, vol. 20. Publications du Laboratoire de Combinatoire et d'Informatique Mathématique, Université du Québec à Montréal (February 1995)
5. Bruyère, V., Hansel, G., Michaux, C., Villemaire, R.: Logic and p-recognizable sets of integers. Bull. Belgian Math. Soc. 1, 191–238 (1994); Corrigendum, Bull. Belg. Math. Soc. 1, 577 (1994)
6. Charlier, É., Rampersad, N., Shallit, J.: Enumeration and Decidable Properties of Automatic Sequences. In: Mauri, G., Leporati, A. (eds.) DLT 2011. LNCS, vol. 6795, pp. 165–179. Springer, Heidelberg (2011)
7. Costa, J.C.: Biinfinite words with maximal recurrent unbordered factors. Theoret. Comput. Sci. 290, 2053–2061 (2003)
8. Currie, J.D., Saari, K.: Least periods of factors of infinite words. RAIRO Inform. Théor. App. 43, 165–178 (2009)
9. Dekking, F.M., Mendès France, M., van der Poorten, A.J.: Folds! Math. Intelligencer 4, 130–138, 173–181, 190–195 (1982), erratum 5 (1983)
10. Duval, J.P.: Une caractérisation de la période d'un mot fini par la longueur de ses facteurs primaires. C. R. Acad. Sci. Paris 290, A359–A361 (1980)
11. Duval, J.P.: Relationship between the period of a finite word and the length of its unbordered segments. Discrete Math. 40, 31–44 (1982)
12. Duval, J.P., Harju, T., Nowotka, D.: Unbordered factors and Lyndon words. Discrete Math. 308, 2261–2264 (2008)
13. Ehrenfeucht, A., Silberger, D.M.: Periodicity and unbordered segments of words. Discrete Math. 26, 101–109 (1979)
14. Glenn, J., Gasarch, W.I.: Implementing WS1S Via Finite Automata. In: Raymond, D.R., Yu, S., Wood, D. (eds.) WIA 1996. LNCS, vol. 1260, pp. 50–63. Springer, Heidelberg (1997)
15. Glenn, J., Gasarch, W.I.: Implementing WS1S Via Finite Automata: Performance Issues. In: Wood, D., Yu, S. (eds.) WIA 1997. LNCS, vol. 1436, pp. 75–86. Springer, Heidelberg (1998)
16. Harju, T., Nowotka, D.: Periodicity and unbordered words: a proof of the extended duval conjecture. J. Assoc. Comput. Mach. 54, 1–20 (2007)
17. Holub, S.: A proof of the extended Duval's conjecture. Theoret. Comput. Sci. 339, 61–67 (2005)

18. Holub, S., Nowotka, D.: On the relation between periodicity and unbordered factors of finite words. Internat. J. Found. Comp. Sci. 21, 633–645 (2010)
19. Honkala, J.: A decision method for the recognizability of sets defined by number systems. RAIRO Inform. Théor. App. 20, 395–403 (1986)
20. Nielsen, P.T.: A note on bifix-free sequences. IEEE Trans. Inform. Theory IT-19, 704–706 (1973)
21. Prodinger, H., Urbanek, F.J.: Infinite 0–1-sequences without long adjacent identical blocks. Discrete Math. 28, 277–289 (1979)
22. Rampersad, N., Shallit, J., Wang, M.W.: Inverse star, borders, and palstars. Inform. Process. Lett. 111, 420–422 (2011)
23. Rudin, W.: Some theorems on Fourier coefficients. Proc. Amer. Math. Soc. 10, 855–859 (1959)
24. Saari, K.: On the Frequency and Periodicity of Infinite Words. Ph.D. thesis, University of Turku, Finland (2008)
25. Shallit, J.: The critical exponent is computable for automatic sequences. In: Ambrož, P., Holub, S., Masáková, Z. (eds.) WORDS 2011: 8th International Conference. Elect. Proc. Theor. Comput. Sci., pp. 231–239 (2011), revised version, with L. Schaeffer, `http://arxiv.org/abs/1104.2303v2`
26. Shapiro, H.S.: Extremal problems for polynomials and power series. Master's thesis, MIT (1952)
27. Thue, A.: Über unendliche Zeichenreihen. Norske vid. Selsk. Skr. Mat. Nat. Kl. 7, 1–22 (1906); reprinted in Nagell, T. (ed.) Selected Mathematical Papers of Axel Thue, pp. 139–158. Universitetsforlaget, Oslo (1977)
28. Thue, A.: Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. Norske vid. Selsk. Skr. Mat. Nat. Kl. 1, 1–67 (1912); reprinted in Nagell, T. (ed.) Selected Mathematical Papers of Axel Thue, pp. 413–478. Universitetsforlaget, Oslo (1977)