Inferring Automatic Sequences

Klaus Sutner[a,*], Sam Tetruashvili[1]

[a] *Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[b] *Google, Mountain View, CA 94043, USA*

Abstract

We give tight upper and lower bounds on the number of terms of an automatic sequence needed to construct the corresponding minimal generating automaton.

Keywords: Automatic sequences, state complexity, machine learning

1. Motivation

Given a sequence of integers, it is a standard problem to try to determine a general and simple description for the whole infinite sequence when only a finite initial segment is available. For example, the computer algebra system *Mathematica* [17] offers operations `FindLinearRecurrence` and `FindGeneratingFunction` that attempt to express the given initial segment as a linear recurrence or to compute a generating function for the sequence. An excellent web-based resource for the identification of integer sequences is Sloane's Online Encyclopedia of Integer Sequences (OEIS), [13]. In this paper we study the subproblem of inferring automatic sequences, sequences that can be described by finite state machines. Classical examples of automatic sequences include the Prouhet-Thue-Morse sequence and the Rudin-Shapiro sequence, see [2, 3].

Since the range of these sequences is necessarily bounded, it is convenient to think of them as infinite words. Thus, an *automatic word* is an infinite word $A \in \Delta^\omega$ over some finite alphabet Δ that can be generated by a finite state machine in the sense that the machine on input n will output $A(n)$. The input alphabet of the machine will be a digit alphabet of the form $\Sigma_k = \{0, 1, \ldots, k-1\}$ and the number n will be represented in *reverse base* k notation. As is shown in [2], we could equally well choose ordinary base k; however, our convention is more appropriate since it allows for the direct computation of candidate machines using decimations and kernels, see section 2.3.

Of course, to formally establish automaticity of an infinite word A we need a general description of A, say, a simple program that computes $A(n)$ given n. As is pointed out in [4], "in practice the following procedure often succeeds in deducing" the more general property of k-regularity "from

*Corresponding author
Email addresses: `sutner@cs.cmu.edu` (Klaus Sutner), `tetruashvili.sam@gmail.com` (Sam Tetruashvili)

knowledge of the first few values." As we shall see, the number of terms required to infer even a k-automatic word whose minimal generating finite state machine has state complexity m may be as large as k^{2m-2}; see also [16] and [12] for similar results. In general, the number of terms required for proper inference is exponential in two natural parameters associated with the minimal recognizer, its depth and discriminating length, see section 2.3 below. We have applied our algorithm to Sloane's OIES, apparently with fairly good accuracy.

2. Kernels and State Complexity

As already pointed out, our default numeration system is reverse base k, in symbols $\mathsf{rrep}_k(n) = d_r d_{r-1} \dots d_0$ where $n = \sum_{i \le r} d_i B^{r-i}$ and $d_r \neq 0$. Let us fix the empty word as the representation for zero but note that we allow trailing zeros. Correspondingly, we write $\mathsf{rval}_k(w)$ for the numerical value of w. To determine $A(n)$ given $\mathsf{rrep}_k(n)$ we consider a slight generalization of ordinary DFAs whose states are labeled by letters in Δ. See [11] for general background.

Definition 2.1 *A* partitioned deterministic finite automaton (PDFA) *is an automaton of the form* $M = \langle\, Q, \Sigma, \delta; q_0, \boldsymbol{F} \,\rangle$ *where* $\langle\, Q, \Sigma, \delta \,\rangle$ *is a deterministic and complete transition system,* $q_0 \in Q$ *an initial state and* $\boldsymbol{F} = (F_a \mid a \in \Delta)$ *is a partition of* Q*. The cardinality of* Δ *is the* order *of* M*.*

We are here mostly interested in the case $\Sigma = \Sigma_k$. It will be convenient to think of the index set Δ of $\boldsymbol{F}$ as an alphabet. This is clearly equivalent to the DFAOs used in [2], but coexists more readily with standard minimization algorithms based on refining given partitions of the state set. For $a \in \Delta$ the *a-behavior* of a state p in M as $[\![p]\!]_a = \{\, x \in \Sigma^\star \mid \delta(p, x) \in F_a \,\}$. The *behavior* of p is the vector $(\, [\![p]\!]_a \mid a \in \Delta \,)$. Slightly abusing terminology, we define the *language* of M as $\mathcal{L}(M) = [\![q_0]\!] \subseteq (\Sigma^\star)^{|\Delta|}$. The automaton is *reduced* if its states all have distinct behavior. It is easy to see that for every PDFA there is an equivalent one that is accessible and reduced. Moreover, this PDFA is unique up to isomorphism. We will refer to this automaton as the *minimal PDFA*.

Definition 2.2 *An infinite word* A *over alphabet* Δ *is* k-automatic *if there exists a PDFA* M *over alphabets* Σ_k *and* Δ *such that for all* $w \in \Sigma_k^\star$*:* $\delta(q_0, w) \in F_{A(\mathsf{rval}_k(w))}$*.*

We will say that M *generates* A to distinguish this situation from other forms of acceptance of infinite words. Thus, our machines operate on reverse base k representations. This is somewhat arbitrary but will be convenient later in our discussion of kernels. Our definition allows for trailing zeros; it is shown in [2] that this definition is quite robust: one may also exclude trailing zeros or one may use standard base k representation instead without changing the class of automatic words.

2.1. Decimation and Kernels

Let $A \in \Delta^\omega$ be an infinite word and α a finite prefix of A, in symbols $\alpha \sqsubset A$. In order to test whether α can plausibly be considered as the prefix of a k-automatic sequence we have to attempt

to construct a machine that generates at least α. The construction ought to succeed and return the minimal PDFA for A whenever A is indeed k-automatic and the prefix $\alpha \sqsubset A$ is sufficiently long. When A fails to be k-automatic the algorithm should indicate that this is likely to be the case. Likewise, when α is too short the algorithm should provide appropriate feedback. In this section we describe an algorithm that has these properties and establish precise bounds for the length of α required to properly identify A.

Since any brute-force enumeration of candidate machines appears to be computationally infeasible, we construct instead a machine directly from the given prefix α. To this end we use the characterization of automaticity based on decimations and kernels.

Definition 2.3 *Let $A : \mathbb{N} \to \Gamma$ be an infinite word. Given a* stride *$s \geq 1$ and an* offset *$d \geq 0$ we can define the* decimation *of A with respect to s and d by $A[s, d](i) = A(s \cdot i + d)$.*

For example, for the well-known Prouhet-Thue-Morse word T [3] we have $T[2, 0] = T$ but $T' = T[2, 1] \neq T$. Moreover, $T'[2, 0] = T'$ and $T'[2, 1] = T$.

Proposition 2.1 *The operation $*$ defined by $[s, d] * [s', d'] = [ss', sd' + d]$ induces a monoid structure on $\mathbb{N}^+ \times \mathbb{N}$ with neutral element $[1, 0]$, a semidirect product of the multiplicative monoid on $\mathbb{N}^+$ and the additive monoid on $\mathbb{N}$. We refer to this monoid as the decimation monoid.*

Proposition 2.2 *The decimation monoid naturally acts on the right on Γ^ω.*

We are interested in the case where the stride s is a power of the base $k \geq 2$. Clearly, the corresponding submonoid is generated by the elements $[k, i]$ where $0 \leq i < k$. Also note that for any $0 \leq n < k^i$ we have $A(n) = \mathsf{fst}(A[k^i, n])$ where fst extracts the first letter of an infinite word. Thus, we can recover the letters of the word from the first letters of the various decimations. This is particularly interesting when the total number of these decimations is finite.

Definition 2.4 *The k-*kernel *of a word $A \in \Gamma^\omega$ is defined by $\mathsf{Ker}_k(U) = \{\, A[k^i, j] \mid 0 \leq i, 0 \leq j < k^i \,\}$*

For example, the 2-kernel of the PMT word T consists of T and $T[2, 1]$. Consider a PDFA M generating a k-automatic sequence A. For the sake of this argument, let us interpret the behavior $[\![p]\!]$ of a state p in M to be the word generated with p as the initial state. Then we have $[\![\delta(p, d)]\!] = [\![p]\!][k, d]$. To see this, recall that M is working on LSD-first representations of numbers. Correspondingly, we can define a right action of $\Sigma_k^\star$ on Γ^ω by $U[w] = U[k^{|w|}, \mathsf{rval}_k(w)]$. Thus $[\![\delta(q_0, w)]\!] = A[w]$ by induction on w. Hence the states in M correspond to the decimations of A with strides k^i.

The following theorem expresses automaticity in terms of kernels [9]. We briefly restate the proof in a form suitable for our purposes.

Theorem 2.1 (Eilenberg) *A sequence is k-automatic if, and only if, its k-kernel is finite.*

Proof. First assume A is k-automatic via a PDFA M on m states. Choose a transfer sequence x_p for every state p in M. As we have seen, every kernel element B is of the form $B = A[w]$ for some word w. But then $B = A[x_{\delta(q_0,w)}]$ and the kernel has cardinality at most m.

For the opposite direction consider the finite kernel K of A. Define a PDFA M by $\langle K, \Sigma_k, \delta, A, \boldsymbol{F} \rangle$ where $\delta(B, a) = B[a]$ and $F_a = \{ B \mid \mathsf{fst}(B) = a \}$. It is easy to see that M has the right properties. □

The size of the minimal PDFA generating an automatic word is a natural measure of its complexity. Write $\mu_k(A)$ for the number of states of the minimal PDFA generating A in reverse base k. By Cobham's celebrated theorem [7], any infinite word A that fails to be ultimately periodic cannot be both k-automatic and ℓ-automatic unless k and ℓ are multiplicatively dependent, i.e., unless $k^i = \ell^j$ for some $i, j > 0$. Let us refer to automatic words that fail to be ultimately periodic as *proper automatic*.

Call $k \geq 2$ *multiplicatively primitive* if k is the least element in its class of multiplicatively dependent numbers. Equivalently, the prime exponents in the factorization of k must have greatest common divisor 1. It is well-known that a word is automatic with respect to base k if, and only if, it is automatic with respect to base k^i.

Proposition 2.3 *$\mu_k(A) \geq \mu_{k^i}(A)$ for all $i \geq 1$.*

To see this, note that we can identify the minimal PDFA for A in base k^i with a subautomaton of the minimal PDFA for base k: consider only states reachable from the initial state under inputs of length a multiple of i. Alas, our measure only accounts for the number of states; the number of transitions may well stay constant even if the number of states decreases in the step from k to k^i. At any rate, for a proper automatic word we can then define the state complexity of A as $\mu_k(A)$ where k is the uniquely determined multiplicatively primitive base for which A is k-automatic.

On the other hand, when A is ultimately periodic it is k-automatic for any base k.

Claim 2.1 *Let $A_m = (12 \ldots m)^\omega$ be the canonical m-periodic word. Then we have:*

- *$\mu_m(A_m) = m + 1$ and $\mu_{m+1}(A_m) = m$,*
- *$\mu_k(A_m) = m\,|S| + b$ for $k \geq m$, S the multiplicative subsemigroup of $\mathbb{Z}_m$ generated by $k \bmod m$ and $b = 0$ when m and k are coprime and 1 otherwise.*

Proof. The first claim is immediate from our definitions.

For the second claim note that we can construct a finite state machine M that generates A_m by forming the closure $Q \subseteq \mathbb{Z}_m \times S$ of $(0, 1)$ under the maps $(p, s) \xrightarrow{a} (p + a \cdot s, s \cdot k) \bmod m$. where $0 \leq a < k$. Since $k \geq m$, it is easy to see that in fact $Q = \mathbb{Z}_m \times S$. The correction factor b is required since $1 \notin S$ when m and k fail to be coprime. The partition is determined by the first components of the states and it is easy to see that M is reduced. □

As a consequence, the sequence $(\mu_k(A_m))_k$ is ultimately periodic with period m. Note that the condition $k \geq m$ is necessary in general, see table 1 which lists $\mu_k(A_p)$ for $2 \leq k, m \leq 20$.

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	6	4	3	6	4	3	6	4	3	6	4	3	6	4	3	6	4	3	6
4	7	8	5	4	9	8	5	4	9	8	5	4	9	8	5	4	9	8	5
5	20	20	10	6	5	20	20	10	6	5	20	20	10	6	5	20	20	10	6
6	13	7	7	12	7	6	13	7	7	12	7	6	13	7	7	12	7	6	13
7	21	42	21	42	14	8	7	21	42	21	42	14	8	7	21	42	21	42	14
8	15	16	13	16	23	16	9	8	25	16	17	16	25	16	9	8	25	16	17
9	54	13	27	54	16	27	18	10	9	54	19	27	54	19	27	18	10	9	54
10	41	40	21	11	11	40	41	20	11	10	41	40	21	11	11	40	41	20	11
11	110	55	55	55	110	110	110	55	22	12	11	110	55	55	55	110	110	110	55
12	27	25	13	24	19	24	25	13	23	24	13	12	37	25	13	24	25	24	25
13	156	39	78	52	156	156	52	39	78	156	26	14	13	156	39	78	52	156	156
14	43	84	43	84	29	15	15	42	85	42	85	28	15	14	43	84	43	84	29
15	60	61	30	31	16	60	60	31	16	30	61	60	30	16	15	60	61	30	31
16	31	64	21	64	55	32	25	32	59	64	29	64	63	32	17	16	65	64	33
17	136	272	68	272	272	272	136	136	272	272	272	68	272	136	34	18	17	136	272
18	109	22	55	108	25	54	37	19	19	108	31	54	109	34	55	36	19	18	109
19	342	342	171	171	171	57	114	171	342	57	114	342	342	342	171	171	38	20	19
20	83	80	41	21	27	80	81	40	31	40	81	80	55	41	21	80	99	40	21

Table 1: The state complexity of minimal PDFAs for periodic sequences up to period and base 20.

The minimal PDFA for a general m-periodic word is a quotient of the minimal PDFA for A_m, so the values provide upper bounds. Note, though, that the exact values for m-periodic words $B_m = (00\ldots01)^\omega$ is quite complicated, see [1] for ordinary base k and [14] fore reverse base k.

2.2. Co-Complexity

Recall that we use reverse base k as the default representation. Needless to say, there is also a minimal PDFA M^{rev} that generates A using standard base k. We refer to the number of states of M^{rev} as the *state co-complexity* of A.

Suppose M is an accessible PDFA. We will show how to construct the minimal PDFA M^{rev} for the (component-wise) reversal of $\mathcal{L}(M)$. Recall Brzozowski's [5] result that automata can be minimized by applying reversal and determinization twice. In our case, M^{rev} is isomorphic to $\mathsf{det}(\mathsf{rev}(M))$ where det denotes determinization using the standard Rabin-Scott algorithm (accessible part only) and rev denotes reversal. We can generalize this construction to PDFAs as follows. Write $M = \langle\, Q, \Sigma, \Delta, \delta; q_0, \boldsymbol{F} \,\rangle$ for the original PDFAs and set

$$
\begin{aligned}
M_a &= \langle\, Q, \Sigma, \delta; q_0, F_a \,\rangle \qquad \text{for } x \in \Delta \\
M'_a &= \mathsf{det}(\mathsf{rev}(M_a)) \\
M' &= \bigoplus_{a \in \Delta} M'_a
\end{aligned}
$$

Attach a partition to M' by setting $F'_a = \{\, \boldsymbol{P} \mid q_0 \in P_a \,\}$

Theorem 2.2 *The PDFA M' is minimal and isomorphic to M^{rev}.*

Proof. First note that by construction $\mathcal{L}(M') = \mathcal{L}(M)^{\mathsf{rev}}$. A simple induction shows that each state $\boldsymbol{P} = \delta'(\boldsymbol{F}, x)$ of M' forms a partition of Q, so suppose that $\boldsymbol{P} \neq \boldsymbol{R}$, say, $P_a \neq R_a$.

We may safely assume that $p \in P_a - R_a$. But M is accessible, so $\delta(q_0, x) = p$ for some word x. It follows that x^{rev} lies in the a-behavior of $\boldsymbol{P}$ but not of $\boldsymbol{R}$. Hence M' is both accessible and reduced. □

It follows that complexity and co-complexity are exponentially related. Alas, there may indeed be an exponential gap between the two measures: As an example, consider the binary word A_r defined by the language $L_r = 0^\star 1\mathbf{2}^\star 1\mathbf{2}^r 10^\star$ in the sense that $A_r(n) = 1$ iff $\mathsf{rrep}_3(n) \in L_r$. Then the state complexity of A_r is $2^{r+2} + 1$, but the co-complexity is $r + 3$.

2.3. The Kernel Algorithm

Suppose we are given a k-automatic word A. How can we compute its state complexity and the corresponding minimal PDFA? For the time being, let us suppose we can directly manipulate infinite words. In particular we need to be able to compute decimations $A[k^i, j]$ and check them for equality. Then we can compute the kernel of A and actually the minimal PDFA using the standard closure algorithm: close $\{A\}$ under the operations $X \mapsto X[k, j]$, $0 \leq j < k$.

```
1    // kernel algorithm
2
3    K = {A};
4    set A active;
5    while( some active B remains )
6           deactivate B;
7           foreach j < k do
8                X = B[k,j];
9                if( X notin K )
10                   add X to K;
11                   set X active;
12               else
13                   add (B,j,Y) to transitions;
```

Note that there are only two non-logical operations in the algorithm:

- decimation (to compute `X` from `B` in line 8),
- equality testing (to check if `X` is in `K` in line 9)

Decimation naturally extends to an operation over finite words. However, the length of the prefix shrinks roughly by a factor of k at each step. To test equality we compare prefixes: $x =_p y$ if $x \sqsubseteq y$

or $y \sqsubseteq x$. Thus, the shorter word has to be a prefix of the longer. Note that $\alpha, \beta \sqsubset A$ implies $\alpha =_p \beta$. For $\alpha \sqsubset A$ we have $\alpha[w] \sqsubset A[w]$, but it may well happen that $\alpha[w] =_p \alpha[v]$ when in fact $A[w] \neq A[v]$. Running the kernel algorithm with these modified non-logical operations will thus in general produce an under-approximation, we may obtain false identities.

We now determine the required number of terms of A to ensure success of the kernel algorithm.

Theorem 2.3 *Let A be k-automatic with state complexity m. The prefix of length k^{2m-3} of A suffices to determine the state complexity, and the prefix of length k^{2m-2} suffices to determine the minimal PDFA. Moreover, both bounds are tight.*

Proof. Suppose the k-kernel of A is $K = \{A = A_1, \ldots, A_m\}$. There are witnesses $v_i \in \Sigma_k^\star$ of length at most $m-1$ such that $A_i = A[v_i]$. Moreover, for $i \neq j$ there are discriminating words $w_{i,j}$ of length at most $m-2$ such that $\mathsf{fst}(A_i[w_{i,j}]) \neq \mathsf{fst}(A_j[w_{i,j}])$. It follows that the kernel algorithm from above will produce the correct number of kernel elements given a prefix of length k^{2m-3}. To obtain the whole PDFA we need the first k^{2m-2} letters.

To see that the bounds are tight consider the regular languages

$$L_1 = \{\, x \in \mathbf{2}^\star \mid \#_1 x = r \pmod{r+1} \,\}$$

$$L_2 = \{\, x \in \mathbf{2}^\star \mid \#_1 x = r \pmod{r+2} \,\}$$

$$L_3 = \{\, x \in \mathbf{2}^\star \mid \#_1 x = r \,\}$$

and write A_1, A_2 and A_3 for the corresponding binary words. A_1 has state complexity $r+1$ and A_2 has state complexity $r+2$ but they agree on their first $2^{2m-3}-1$ bits. Words A_2 and A_3 both have kernels of size $m = r+2$ but distinct PDFAs, yet they agree on their first $2^{2m-2}-1$ bits. □

It is interesting to study the lower bound examples in the last theorem more carefully. For $r = 5$, the kernel algorithm applied to the first 2^{11} bits of A_2 produces the machine in figure 1. The machine is correct except that the transition labeled 0 with source state 7 should be a self-loop.

In general the two critical parameters that determine the number of terms required in the inference algorithm are the following. First, the *depth* of the minimal PDFA, i.e., the length of the longest path from the initial state to an arbitrary state. Second, the length of the longest *discriminating string* that differentiates the between the behavior of two distinct states.

As an example, consider the square-free bi-infinite ternary sequence introduced by Kurosaki in [10]. The bi-infinite sequence is obtained by iterating the map $\varphi(x) = \overline{x}\, x\, \widehat{x}$ where $\overline{x}$ is defined by the substitution $x[0 \mapsto 1, 1 \mapsto 0]$ and $\widehat{x}$ by $x[0 \mapsto 2, 2 \mapsto 0]$. From this we can derive a morphism f and a substitution g as follows:

$$f : (1 \to 123, 2 \to 214, 3 \to 541, 4 \to 632, 5 \to 365, 6 \to 456)$$

$$g : (1 \to 0, 2 \to 1, 3 \to 2, 4 \to 2, 5 \to 1, 6 \to 0)$$

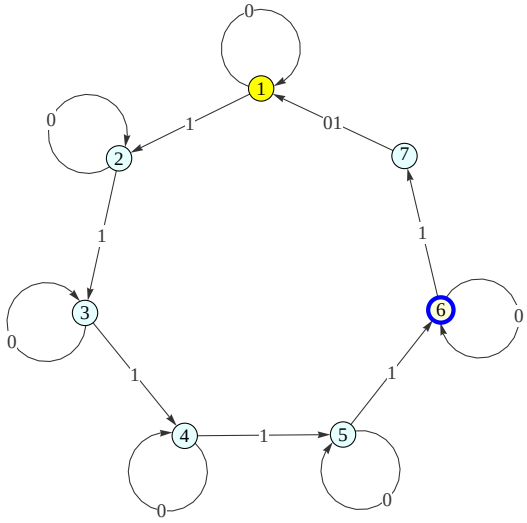

Figure 1: The machine produced by the kernel algorithm on input the first 2^{11} bits of A_2.

Then $g(\lim f^n(1))$ is a square-free infinite word K that starts like so:

$$0121021201020120212010210121020120210121021202101201021202102 0$$

The morphism and substitution together determine a PDFA that generates K in ordinary base 3 notation and to co-complexity of K turns out to be 6. The state complexity of K is also 6 and the PDFA constructed by the algorithm from the prefix of length 81 is shown in figure 2.

The inference algorithm uses the kernel algorithm, possibly repeatedly, to ascertain automaticity of a given finite sequence α.

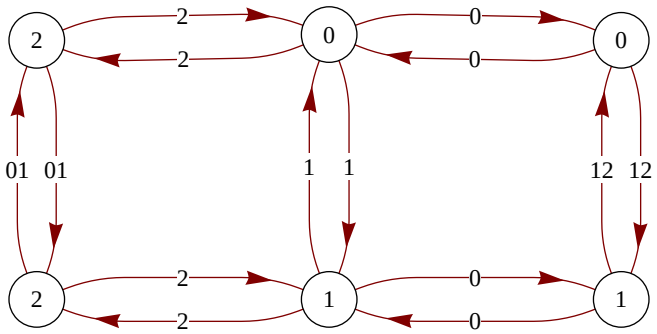

Figure 2: The minimal PDFA for the Kurosaki word. State labels represent the partition.

3. Inferring Automaticity

We have used our algorithm to test the sequences of the Online Encyclopedia of Integer Sequences [13]. Note that there is no easy way to determine whether a sequence in OEIS is infinite or finite, so we use heuristics to weed out sequences that are too short. We also omit sequences that would require large alphabets and may be assumed to have infinite range. The remaining sequences are recoded as words over a suitable digit alphabet Δ_d. Next we try to determine whether A is ultimately periodic by checking whether all sufficiently long prefixes of A end in a cube, a condition that is known to imply ultimate periodicity in the infinite case, see [6].

In the interesting case when A is not found to be ultimately periodic the algorithm picks a proper prefix $\alpha \sqsubset A$ and attempts to compute the k-kernel of α for various small multiplicatively primitive values of k. The computation is deemed successful when the automaton based on α properly generates all of A. In the experiment, we used our algorithm as part of a larger package that attempts to infer integer sequences, see [15]. As of April 30, 2010, the database contained 175117 integer sequences. Of these, the package inferred a total of 33157 with high confidence and identified 1828 sequences as automatic.

Acknowledgments: We are grateful to Jeff Shallit for provinding references [12] and [16], and to Neil Sloane for discovering errors in an earlier version of table 1.

References

[1] B. Alexeev. Minimal DFA for testing divisibility. *J. Comput. System Sci.*, 69(2):235–243, 2004.

[2] J.-P. Allouche and J. Shallit. *Automatic Sequences*. Cambridge UP, 2003.

[3] J.-P. Allouche and J. O. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In Ding et al. [8], pages 1–16.

[4] J.-P. Allouche and J. O. Shallit. The ring of k-regular sequences, II. *Theoretical Computer Science*, 307(1):3–29, 2003. WORDS.

[5] J. A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical Theory of Automata*, volume 12 of *MRI Symposia Series*, pages 529–561. Polytechnic Press, Polytechnic Institute Brooklyn, 1963.

[6] C. Choffrut and J. Karhumäki. Combinatorics of words. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 6. Springer Verlag, 1997.

[7] A. Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Math. Systems Theory*, 6:186–192, 1972.

[8] C. Ding, T. Helleseth, and H. Niederreiter, editors. *Sequences and Their Applications*. Springer, Berlin, 1999.

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

[10] T. Kurosaki. Direct definition of a ternary infinite square-free sequence. *Information Processing Letters*, 106:175–179, 2008.

[11] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[12] J. Shallit. Remarks on inferring integer sequences. `https://cs.uwaterloo.ca/~shallit/Talks/infer.ps`, 1995.

[13] N. J. A. Sloane. Online encyclopedia of integer sequences. `http://www.research.att.com/~njas/sequences/`.

[14] K. Sutner. Divisibility and state-complexity. *The Mathematica Journal*, 11(3):430–445, 2009.

[15] S. Tetruashvili. Inductive inference of integer sequences. Sen. Thesis, Carnegie Mellon U., Advisor: M. Blum, Mai 2010.

[16] L. P. J. Veelenturf. Inference of sequential machines from sample computations. *IEEE Trans. Computers*, 27:167–170, 1978.

[17] S. Wolfram. Mathematica. `http://www.wolfram.com/`.