The Logical Approach to Automatic Sequences
Part 4: Enumeration and Automatic Sequences

Jeffrey Shallit
School of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
`shallit@cs.uwaterloo.ca`
`https://cs.uwaterloo.ca/~shallit`

Main points of today's talk

- Many quantities dealing with k-automatic sequences are k-regular
- Many quantities dealing with automatic sequences can be "automatically" enumerated
  - That is, we can algorithmically construct a polynomial-time algorithm to enumerate the quantity, given a first-order formula describing it
- We should add this to our "combinatorial arsenal" of techniques, along with more traditional enumeration decision methods (Wilf, Gosper, Zeilberger, etc.)
- In between k-automatic and k-regular is *k-synchronized*, which gives even faster algorithms for enumeration in some cases

What is a formula?

- Traditional (vague) answer: an expression involving traditional operation such as addition, subtraction, multiplication, division, exponentiation, and perhaps additional functions such as factorial, binomial coefficient, trig and inverse trig functions, n'th roots, logarithm, floor, ceiling, summation, product, special functions, and so forth.
- Modern (precise) answer: an enumeration formula is an algorithm that runs in little-o of the time required to actually list the things being enumerated.
- A *good* formula is one that runs in time polynomial in n and the output size.
- A *very good* formula is one that runs in time polynomial in $\log n$ and the output size.
- See Herb Wilf, What is an answer?, *Amer. Math. Monthly* **89** (1982), 289–292.

Enumerating aspects of automatic sequences

Many papers in the literature are concerned with enumerating various aspects of automatic sequences.

For example: *subword complexity* $\rho(n)$, the number of distinct factors of a sequence of length n.

A classic result of Cobham: for automatic sequences $\rho(n) = O(n)$.

For morphic sequences it is possible for the subword complexity to be as high as $\Omega(n^2)$: for example, the fixed point, starting with a, of the morphism $a \to ab$, $b \to bc$, and $c \to c$:

$$abbcbccbcccbccccbcccccbccccccbcccccccbccccccccbccccccccc\cdots$$

Other aspects of enumeration

Palindrome complexity: the number of distinct factors of length n that are palindromes. (See Allouche, Baake, et al., 2003)

Unbordered complexity: the number of distinct factors of length n that are unbordered. (See Currie and Saari 2009)

Reversal complexity: the number of distinct factors of length n whose reversals are also factors.

Conjugate complexity: the number of distinct factors of length n whose conjugates are also factors.

Squares: the number of distinct factors of length $2n$ that are squares of order n. (Order of square xx is $|x|$.)

The punch line

If P is a property of the factors of a k-automatic sequence that is expressible in a first-order formula, then the number $f_P(n)$ of such length-n factors is k-regular.

This means there is a linear representation, in terms of matrices and vectors for computing $f_P(n)$.

This always gives an algorithm A_P to compute $f_P(n)$ that runs in $O((\log n)^3)$ time: a very good formula!

However:

- Given P, *finding* the algorithm A_P might take a huge amount of time (depending on P)
- the constant factor in the $O((\log n)^3)$-time algorithm might be ridiculously large.

Nondeterministic automata and path cardinalities

Theorem. Given an NFA $M = (Q, \Sigma, \delta, q_0, F)$ define a matrix-valued morphism

$$\mu(a)_{i,j} = \text{number of paths labeled } a \text{ from } q_i \text{ to } q_j.$$

Then for words x the quantity $\mu(x)_{i,j}$ is the number of paths labeled x from q_i to q_j.

Proof. By induction on the length of the path.

Corollary. Let $v = [1\ 0\ 0\ \cdots\ 0]$, where there is 1 in the position of the start state q_0, and w^T a boolean vector with 1's in positions of the final states. Then $v\mu(x)w^T > 0$ if and only if x is accepted by M.

k-regular sequences and enumeration

k-regular sequences and their connections to automata give a framework for enumerating these aspects of automatic sequences.

Basic idea:

Theorem. Let $S \subseteq \mathbb{N} \times \mathbb{N}$. Given a DFA accepting the language

$$L = \{(i, n)_k \ : \ (i, n) \in S\},$$

the function

$$f_S(n) = |\{i \ : \ (i, n) \in S\}|$$

is k-regular.

Proof. From the DFA M accepting L, make an NFA M' by projecting each label on a transition to its second coordinate. Thus, for example, transitions labeled $[0, 1]$ and $[1, 1]$ from q_i to q_j get projected to two arrows labeled 1 from q_i to q_j.

Now use the theorem about NFA's.

An example: counting palindromic factors

How do we count, for example, the palindromic factors of length n — call it $f(n)$ — of an automatic sequence such as **t** ?

An obvious first try is to consider the language

$$L_{\text{pal}} = \{(i,n)_2 \ : \ S[i..i+n-1] \text{ is a palindrome } \}.$$

However, this doesn't work because each n can have many different i associated with it, and we are double- (or triple- or more) counting the same palindrome many times.

What we need to do is count a single i for each distinct palindrome. The easiest way to do this is to count, not occurrences of palindromes, but *first occurrences* of palindromes in the sequence.

Thus what we *really* want is

$$L_{\text{pal}} = \{(i,n)_2 \ : \ S[i..i+n-1] \text{ is a palindrome and any occurrence of the same factor } S[j..j+n-1] \text{ has } j \geq i\}$$

Counting palindrome factors

We can do this with the first-order formula

$$(\forall l\ (l < n) \Rightarrow S[i+l] = S[(i+n)-(l+1)]) \\ \wedge\ (\forall j\ (\forall m\ (m < n) \Rightarrow S[i+m] = S[j+m]) \Rightarrow (j \geq i)).$$

In `Walnut`, for the Thue-Morse sequence, this is:

```
eval tmpalc "(Al (l<n) => T[i+l] = T[(i+n)-(l+1)]) &
    (Aj (Am (m<n) => T[i+m] = T[j+m]) => (j>=i))":
```

Getting the matrices from Walnut

The latest version of Walnut allows one to obtain the matrices (in Maple format) corresponding to a formula. The syntax is

```
eval tmpalc n "(Al (l<n) => T[i+l] = T[(i+n)-(l+1)]) &
    (Aj (Am (m<n) => T[i+m] = T[j+m]) => (j>=i))":
```

Here "n" can be replaced by any list of free variables.

The result is stored (in this case) in the file tmpalc.mpl.

Palindrome complexity of Thue-Morse

We find, for $f(n)$ the number of palindromes of length n of the Thue-Morse sequence

$$v = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mu(0) = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\quad
\mu(1) = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$w = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$$

Palindrome complexity of Thue-Morse

When we compute the first few terms of the palindrome complexity using these matrices we find

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$f(n)$	1	2	2	2	2	0	4	0	4	0	4	0	4	0	2	0	2	0	4	0	4

This data suggests that the palindrome complexity for **t** is bounded above by 4.

How could we prove this?

The semigroup trick

We know that

$$f(n) = v\mu((n)_2)w$$

for the vectors v, w and matrices $\mu(0), \mu(1)$.

We can compute the size of the semigroup generated by $\mu(0)$ and $\mu(1)$ using a queue-based algorithm.

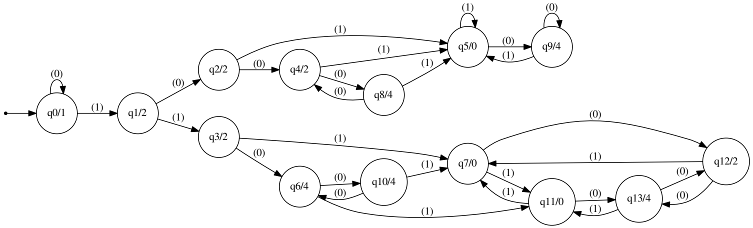
It is 68.

By computing vMw for all M in this semigroup, we see that $f(n) \in \{0, 1, 2, 4\}$.

The semigroup trick

Now we can contruct a 2-DFAO out of all possible products Mw, with output of each state Mw equal to vMw.

When we do this, we get an automaton with 68 states that can be minimized to one with 14 states:



Minimal linear representations

There is an algorithm to minimize linear representations.

The result is a representation of smallest rank.

It may not be unique.

For example, when we run it on the matrices for palindrome complexity of Thue-Morse we get the following minimized representation:

$$v' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mu'(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{6}{5} & \frac{1}{5} & \frac{1}{5} & -\frac{7}{10} & \frac{1}{5} \\ 0 & 0 & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & -\frac{2}{5} & \frac{2}{5} \\ 0 & 0 & \frac{1}{5} & \frac{6}{5} & \frac{1}{5} & -\frac{7}{10} & \frac{1}{5} \end{bmatrix} \quad \mu'(1) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{2}{5} & -\frac{3}{5} & -\frac{3}{5} & \frac{11}{10} & \frac{2}{5} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{2}{5} & \frac{3}{5} & \frac{3}{5} & \frac{2}{5} & -\frac{2}{5} \end{bmatrix}$$

$$w' = \begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 0 & 4 \end{bmatrix}^T$$

Finding relations from the matrices

The function $f(n)$ for palindrome complexity of Thue-Morse satisfies

$$\begin{aligned}
f(4n+3) &= f(4n+1)\\
f(8n) &= f(2n)+f(2n+1)-f(4n+1)\\
f(8n+1) &= f(4n+1)\\
f(8n+4) &= f(8n+2)\\
f(8n+5) &= 0\\
f(16n+6) &= f(4n+1)+f(4n+2)\\
f(16n+10) &= f(4n+1)+f(4n+2)\\
f(16n+14) &= f(4n+2)\\
f(32n+2) &= f(8n+2)\\
f(32n+18) &= f(8n+6)
\end{aligned}$$

Determining the relations from the matrices

We can mechanically *find* the relations for *any* given k-regular sequence g.

Suppose we are given the linear representation of a k-regular sequence g, that is, vectors v, w and matrices $M_0, M_1, \ldots, M_{k-1}$ such that

$$g(n) = vM_{a_1} M_{a_2} \cdots M_{a_j} w,$$

where $a_1 a_2 \cdots a_j = (n)_k$.

To make this really work perfectly you need to first insure that $vM_0 = v$. But if you are willing to give up the relations at $n = 0$ this is not absolutely necessary. (This is because of the "leading zeroes" problem; if the canonical representation for n is x, then the canonical representation for $2n$ is $x0$ — *except* if $n = 0$, when it is just x. So if $vM_0 w \neq vw$, you have a small problem.)

Determining the relations from the matrices

Now let M be arbitrary and consider vM as a vector with variable entries, say $[a_1, a_2, \ldots, a_d]$.

Successively compute $vMM_y w$ for words y of length $0, 1, 2, \ldots$ over $\Sigma_k = \{0, 1, \ldots, k-1\}$; this will give an expression in terms of the variables $a_1, \ldots, a_d$.

After at most $d+1$ such relations, we find an expression for $vMM_y w$ for some y as a linear combination of previously computed expressions.

When this happens, you no longer need to consider any expression having y as a suffix.

Eventually the procedure halts, and this corresponds to a system of equations for g.

Determining the relations from the matrices

Example: Let $k = 2$, $v = [6, 1]$, $w = [2, 4]^T$, and

$$\begin{aligned} M_0 &= \begin{bmatrix} -3 & 1 \\ 1 & 4 \end{bmatrix} \\ M_1 &= \begin{bmatrix} 0 & 2 \\ -3 & 1 \end{bmatrix} \end{aligned}$$

Suppose M is some product of M_0 and M_1, and suppose $vM = [a, b]$. We find

$$\begin{aligned} vMw &= 2a + 4b \\ vMM_0w &= -2a + 18b \\ vMM_1w &= -8a - 2b \\ vMM_0M_0w &= 24a + 70b \\ vMM_1M_0w &= 36a + 24b \end{aligned}$$

Determining the relations from the matrices

Solving the linear system, we get

$$\begin{aligned} vMM_1w &= \frac{35}{11}vMw - \frac{9}{11}vM_0w \\ vMM_0M_0w &= 13vMw + vM_0w \\ vMM_1M_0w &= \frac{174}{11}vMw - \frac{24}{11}vM_0w. \end{aligned}$$

This gives us, for $n \geq 1$, that

$$\begin{aligned} g(2n+1) &= \frac{35}{11}g(n) + \frac{9}{11}g(2n) \\ g(4n) &= 13g(n) + g(2n) \\ g(4n+2) &= \frac{174}{11}g(n) - \frac{24}{11}g(2n) \end{aligned}$$

In practice this could be speeded up by not letting vM be completely symbolic, but computing the transitive closure of $T := (M_0 \text{ OR } M_1)$ and putting 0's in the entries that correspond to 0's in T.

Counting palindrome occurrences in Thue-Morse prefixes

A small variation of our technique allows us to compute the number $g(n)$ of nonempty palindromes (not necessarily distinct) occurring in prefixes of length n of $\mathbf{t}$.

We need a formula asserting that a palindrome occurs in a prefix of length n:

$$(i + \ell \leq n) \ \wedge \ (\forall j < l \ \mathbf{t}[i + j] = \mathbf{t}[i + l - (j + 1)]).$$

In Walnut this is

```
eval palpref n "(l>0) & (i+l<=n) & (Aj (j<l) =>
               (T[i+j] = T[i+l-(j+1)]))":
```

The result is a linear representation of rank 29.

It can be minimized to a linear representation of rank 9 on the next slide.

Counting palindrome occurrences in Thue-Morse prefixes

The linear representation for $g(n)$ is given by

$$v = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mu(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & -3 & -1 & 3 & 1 & 0 & -4 & 3 \\ 0 & 4 & -7 & 1 & 2 & 3 & 3 & -10 & 5 \\ 0 & 10 & -17 & 2 & 5 & 8 & 6 & -24 & 11 \end{bmatrix} \quad \mu(1) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & 1 & -2 & 1 & 0 & 1 & 0 & -2 & 2 \\ 0 & 3 & -5 & 0 & 2 & 3 & 1 & -7 & 4 \\ 0 & 5 & -9 & 2 & 2 & 4 & 3 & -12 & 6 \\ 0 & 11 & -18 & 2 & 4 & 10 & 6 & -26 & 12 \end{bmatrix}$$

$$w = \begin{bmatrix} 0 & 1 & 2 & 4 & 6 & 8 & 10 & 12 & 18 \end{bmatrix}^T$$

If we want to get asymptotics for $g(n)$, we can consider n of the form 2^k.

This corresponds to understanding the asymptotics of the entries of $v\mu(1)\mu(0)^k w$.

For this it suffices to understand the asymptotics of $\mu(0)^k$.

Counting palindrome occurrences in Thue-Morse prefixes

Since $\mu(0)$ satisfies its own minimal polynomial $p(X)$, the entries of $\mu(0)^k$ all satisfy a linear recurrence of order at most 9, which can be deduced from $p(X)$.

In Maple the minimal polynomial can be computed with the commands

```
with(linalg);
factor(minpoly(m0,X));
```

and we get

$$p(X) = (X+2)(X-2)^2(X-1)^3(X+1)^3.$$

Counting palindrome occurrences in Thue-Morse prefixes

From the fundamental theorem of linear recurrences this means that $v\mu(1)\mu(0)^k w$ can be expressed in the form

$$c_1(-2)^k + (c_2 + c_3 k)2^k + c_4 + c_5 k + c_6 k^2 + (c_7 + c_8 k + c_9 k^2)(-1)^k.$$

When we solve for the constants, we get

$$c_1 = 1/24 \quad c_2 = 37/72 \quad c_3 = 5/12 \quad c_4 = 1/3$$
$$c_5 = 0 \quad c_6 = 0 \quad c_7 = 1/9 \quad c_8 = 0 \quad c_9 = 0$$

$$\text{so } g(2^k) = \frac{1}{24}(-2)^k + \frac{37}{72}2^k + \frac{1}{3} + \frac{5}{12}k2^k + \frac{1}{9}(-1)^k.$$

This gives the asymptotics of $g(2^k)$ as $\Theta(k2^k)$, and so $g(n) = \Theta(n \log n)$.

In general, more detailed asymptotics may require understanding the *joint spectral radius*, which is not easy to compute.

Counting palindrome occurrences in Thue-Morse prefixes

Exercise: Do the same thing for counting the number $g'(n)$ of *distinct* palindromes occurring in a prefix of length n of the Thue-Morse sequence.

– Find the formula

– Find a linear representation

– Find a closed form for $g'(2^k)$

Summary of results provable with the method

If $\mathbf{a} = (a_n)_{n \geq 0}$ is a k-automatic sequence, then the following associated sequences are (effectively) k-regular.

- its subword complexity function, $n \rightarrow$ number of distinct factors of length n
  - Previously known for fixed points of k-uniform morphisms (Mossé, 1996)
- its palindrome complexity function, $n \rightarrow$ number of distinct factors of length n that are palindromes
  - Previously known for fixed points of primitive k-uniform morphisms (Allouche, Baake, Cassaigne, Damanik, 2003)
- its sequence of separator lengths (length of smallest factor that begins at position n and does not occur previously)
  - Previously known for fixed points of k-uniform circular morphisms (Garel, 1997)

Summary of results

If $\mathbf{a} = (a_n)_{n\geq 0}$ is a k-automatic sequence, then the following associated sequences are k-regular sequences:

- the number of distinct square factors of length n; the number of squares beginning at (centered at, ending at) position n; the length of the longest square beginning at (centered at, ending at) position n; the number of palindromes beginning at (centered at, ending at) position n; the number of distinct recurrent factors of length n; etc.,
  - Previously known for the Thue-Morse sequence (Brown, Rampersad, Shallit, Vasiga, 2006)

New results

If $(a_n)_{n\geq 0}$ is a k-automatic sequence, then the following associated sequences are k-regular sequences:

- The recurrence function of $\mathbf{a}$, $n \to$ the smallest integer t such that every factor of length t of $\mathbf{a}$ contains every factor of length n
- The appearance function of $\mathbf{a}$, $n \to$ the smallest integer t such that the prefix of length t of $\mathbf{a}$ contains every factor of length n

Other computable functions

- Given a regular language L encoding a set S of pairs of integers, the quantity $\sup_{(p,q)\in S} \frac{p}{q}$ is either infinite or rational, and it is computable
- The critical exponent of an automatic sequence (exponent of the largest power of any factor) is a rational number and is computable.
- The optimal constant for linear recurrence for an automatic sequence is rational and computable.

Linear recurrence

A sequence $\mathbf{a} = (a_n)_{n \geq 0}$ is linearly recurrent if there is a constant C such that for all $\ell \geq 0$, and all factors x of length ℓ occurring in $\mathbf{a}$, any two consecutive occurrences of x are separated by at most $C\ell$ positions.

Given $\mathbf{a}$, can we determine the smallest value of C that works?

The idea is, given the automaton for $\mathbf{a}$, to construct an automaton accepting the language of pairs (d, ℓ) such that

(a) there is some factor of length ℓ for which there is another occurrence at distance d and

(b) this occurrence is actually the very next occurrence.

Then $\sup_{(d,\ell) \in S} \frac{d}{\ell}$ gives the optimal C.

Synchronized sequences

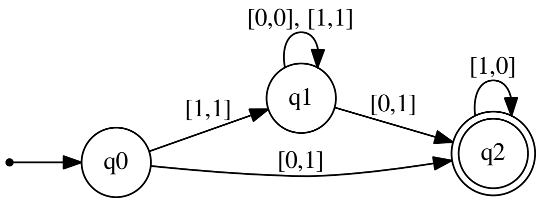
In some cases, the sequence we are trying to enumerate, in addition to being k-regular, has a stronger property: it is *k-synchronized*.

A sequence $(s_n)_{n\geq 0}$ over $\mathbb{N}$ is k-synchronized if there is a DFA accepting the language

$$L = \{(n, s_n)_k \ : \ n \geq 0\}.$$

Synchronized sequences

Example. The function $f(n) = n + 1$ is k-synchronized. For example, for $k = 2$, the following automaton suffices:



Why synchronized sequences?

- If $f(n)$ is k-synchronized, then
  we can compute $f(n)$ in $O(\log n)$ time
- If $f(n)$ is k-synchronized, then $f(n) = O(n)$
- If $f(n)$ is non-decreasing and k-synchronized, then either
  $f(n) = \Theta(1)$ or $f(n) = \Theta(n)$

Efficient computation of synchronized sequences

To compute $f(n)$ in $O(\log n)$ time:

- On input n, construct the $O(\log n)$-state machine M' accepting those words with first component of the form $0^*(n)_k$ and second component anything
- Intersect, using the familiar direct product construction, with the DFA M accepting $\{ \ (n, f(n))_k \ : \ n \geq 0\}$
  - Resulting automaton accepts exactly one word
  - Find accepting path using depth-first search
  - Label of accepting path gives $f(n)$ in base k

Growth bounds

Theorem. If $f(n)$ is k-synchronized, then $f(n) = O(n)$.

Proof.

- Suppose f is k-synchronized and accepted by a DFA with t states.
- If $f(n) \neq O(n)$, then there exists n such that $f(n) > k^t n$.
- So the base-k representation of $(n, f(n))$ starts with at least t zeros in the first component, and a nonzero symbol in the second component.
- Apply the pumping lemma to $z = (n, f(n))_k$
- We see that "pumping" gives a new word in the language with n unchanged, but $f(n)$ increased.
- But f is a function, a contradiction. ■

Closure properties of synchronized sequences

The class of k-synchronized sequences is closed under

- sum
- $\mathbb{N}$-linear combination
- $f(n) \to \lfloor \alpha f(n) \rfloor$ for α rational
- term-wise maximum and minimum
- running maximum: $g(n) = \max_{0 \le i < n} f(i)$
- discrete inverse: $g(n) = \min\{i \ : f(i) \ge n\}$
- composition

Many aspects of k-automatic sequences are k-synchronized

Example: the appearance function.

$A_{\mathbf{x}}(n) =$ length of shortest prefix of $\mathbf{x}$ containing all length-n factors of $\mathbf{x}$

$=$ the smallest integer t such that every length-n factor of $\mathbf{x}$ occurs at least once in $\mathbf{x}[0..t-1]$.

$=$ t such that every length-n factor of $\mathbf{x}$ occurs in $\mathbf{x}[0..t-1]$ but the length-n factor ending at position $t-1$ occurs exactly once in $\mathbf{x}[0..t-1]$

Appearance function formula

$$\begin{aligned} L = \{(n,t)_k \quad : \quad & \forall\, i \geq 0 \ \exists j \leq t-n \\ & \text{such that } \mathbf{x}[i..i+n-1] = \mathbf{x}[j..j+n-1] \\ & \text{and } \forall\, \ell < t-n \\ & \mathbf{x}[\ell..\ell+n-1] \neq \mathbf{x}[t-n..t-1]\}. \end{aligned}$$

Other synchronized functions

- **separator function**: length of the shortest factor of $\mathbf{x}$ beginning at position n that never appeared previously in $\mathbf{x}$ (Carpi & Maggi, 2001)
- **repetitivity index**: the minimal distance between two consecutive occurrences of the same length-n factor in $\mathbf{x}$ (Carpi & D'Alonzo, 2009)
- **recurrence function**: size of the smallest "window" always guaranteed to contain all length-n factors in $\mathbf{x}$ (Charlier & Rampersad & S, 2011)

Subword complexity

$\rho_{\mathbf{x}}(n) =$ number of distinct length-n factors of $\mathbf{x}$

- known to be k-regular
- known to be $O(n)$ for k-automatic sequences
- this suggests it could be k-synchronized

Novel occurrences

Call a length-n factor *novel* at position i if it occurs there but in no earlier location.

Here is a formula for novel factors:

$$\{(n,i)_k \ : \ \forall j, 0 \leq j < i \quad \mathbf{x}[i..i+n-1] \neq \mathbf{x}[j..j+n-1]\}$$

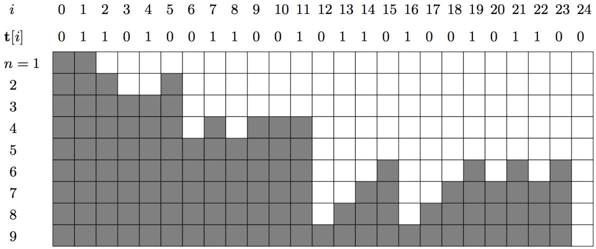
Theorem. In any sequence of linear complexity, the starting positions of novel occurrences of factors are “clumped together” in a bounded number of contiguous blocks.

Novel factors for Thue-Morse

Consider the Thue-Morse sequence

$$\mathbf{t} = t_0 t_1 t_2 \cdots = 0110100110010110 \cdots ,$$

The gray squares in the rows below depict the evolution of novel length-n factors in the Thue-Morse sequence for $1 \leq n \leq 9$.



i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$\mathbf{t}[i]$	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	0
$n = 1$	■	■																							
2	■	■	■			■																			
3	■	■	■	■	■	■																			
4	■	■	■	■	■	■		■		■	■	■													
5	■	■	■	■	■	■	■	■	■	■	■	■													
6	■	■	■	■	■	■	■	■	■	■	■	■				■				■		■		■	
7	■	■	■	■	■	■	■	■	■	■	■	■			■	■			■	■	■	■	■	■	
8	■	■	■	■	■	■	■	■	■	■	■	■		■	■	■		■	■	■	■	■	■	■	
9	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	

Bound on number of contiguous blocks

Theorem. Let $\mathbf{x}$ be an infinite word. For $n \geq 1$, the number of contiguous blocks of starting occurrences of novel factors in row n is at most $\rho_{\mathbf{x}}(n) - \rho_{\mathbf{x}}(n-1) + 1$.
Proof. By induction on n. Base case easy.

Assume true for $n - 1$. We prove for n.

Every position marking the start of a novel occurrence is still novel. Further, in every block except the first, we get novel occurrences at one position to the left of the beginning of the block.

So if row $n - 1$ has t contiguous blocks, then we get $t - 1$ novel occurrences at the beginning of each block, except the first.

Bound on number of contiguous blocks

The remaining $\rho_{\mathbf{x}}(n) - \rho_{\mathbf{x}}(n-1) - (t-1)$ novel occurrences could be, in the worst case, in their own individual contiguous blocks.

Thus row n has at most $t + \rho_{\mathbf{x}}(n) - \rho_{\mathbf{x}}(n-1) - (t-1)$
$= \rho_{\mathbf{x}}(n) - \rho_{\mathbf{x}}(n-1) + 1$ contiguous blocks.

Bound for Thue-Morse

For Thue-Morse example, it is well-known that $\rho_{\mathbf{t}}(n) - \rho_{\mathbf{t}}(n-1) \leq 4$.

So the number of contiguous blocks of novel factors is at most 5.

This is achieved, for example, for $n = 6$.

Linear complexity

Corollary. If the sequence **x** has linear complexity (that is, $\rho_{\mathbf{x}}(n) = O(n)$), then there is a constant C such that every row in the evolution of novel occurrences consists of at most C contiguous blocks.

Proof. By a deep result of Cassaigne (1996) we know that there exists a constant C such that $\rho_{\mathbf{x}}(n) - \rho_{\mathbf{x}}(n-1) \leq C - 1$. Hence from our result, there are at most C contiguous blocks in any row.

Subword complexity of automatic sequences is k-synchronized

Theorem. Let $\mathbf{x}$ be a k-automatic sequence. Then its subword complexity function $\rho_{\mathbf{x}}(n)$ is k-synchronized.

Proof. Construct a DFA to accept
$\{(n, m)_k \ : \ n \geq 0 \text{ and } m = \rho_{\mathbf{x}}(n)\}$.

There is a finite constant $C \geq 1$ such that the number of contiguous blocks of novel factors is bounded by C.

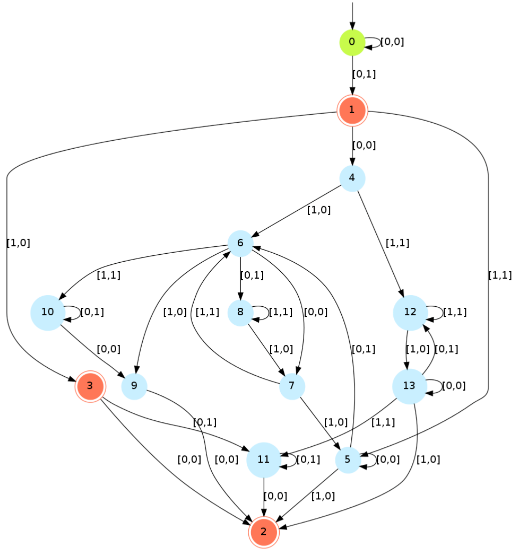
Nondeterministically "guess" the endpoints of every block and then verify that each factor of length n starting at the positions inside blocks is a novel occurrence, while all other factors are not.

Finally, verify that m is the sum of the sizes of the blocks.

A corollary

Corollary. Given a k-automatic sequence $\mathbf{x}$, there is an algorithm that, on input n in base k, will compute the subword complexity $\rho_{\mathbf{x}}(n)$ expressed in base k in time $O(\log n)$.

Example: subword complexity for Thue-Morse



Applications: an improvement on Goldstein

Corollary. There is an algorithm, that, given a k-automatic sequence $\mathbf{x}$, will compute

- $\sup_{n\geq 1} \rho_{\mathbf{x}}(n)/n$,
- $\limsup_{n\geq 1} \rho_{\mathbf{x}}(n)/n$,
- $\inf_{n\geq 1} \rho_{\mathbf{x}}(n)/n$, and
- $\liminf_{n\geq 1} \rho_{\mathbf{x}}(n)/n$.

Proof. We already showed how to construct an automaton accepting $\{(n, \rho_{\mathbf{x}}(n))_k \ : \ n \geq 1\}$. Using Schaeffer & S (2012), we can compute the sup, lim sup etc.

Results

Theorem. If $\mathbf{x}$ is k-automatic, then the following are k-synchronized:

- the function counting the number of distinct length-n factors that are powers;
- the function counting the number of distinct length-n factors that are primitive words.

Sketch of proof

Main ideas:

- A word x is a power if and only if there exist nonempty words y, z such that $x = yz = zy$.
- Thus, we can express the formula $P(i,j) :=$ "$\mathbf{x}[i..j]$ is a power" as follows: "there exists d, $0 < d < j - i + 1$, such that $\mathbf{x}[i..j-d] = \mathbf{x}[i+d..j]$ and $\mathbf{x}[j-d+1..j] = \mathbf{x}[i..i+d-1]$".
- Furthermore, we can express the formula $P'(i,n) :=$ "$\mathbf{x}[i..i+n-1]$ is a length-n power and is a novel occurrence of that factor in $\mathbf{x}$".
- We show that once again the novel occurrences of length-n powers are clustered into a finite number of blocks.

Sketch of proof

- Then we can nondeterministically guess the endpoints of these blocks, and verify that the length-n factors beginning at the positions inside the blocks are novel occurrences of powers, while those outside are not, and sum the lengths of the blocks, using a finite automaton built from M.
- So the counting function for powers is k-synchronized.
- The number of length-n primitive words in $\mathbf{x}$ is then also k-synchronized, since it is expressible as the total number of words of length n, minus the number of length-n powers.

Unsynchronized sequences

Are other aspects of k-automatic sequences always k-synchronized?

No.

Recall that a word w is *bordered* if it has a nonempty prefix, other than w itself, that is also a suffix. Alternatively, w is bordered if it can be written in the form $w = tvt$, where t is nonempty. Otherwise a word is *unbordered*.

Unsynchronized sequences

Theorem. The characteristic sequence of the powers of 2 $\mathbf{c} = 0110100010\cdots$ is 2-automatic, but the function $u_{\mathbf{c}}(n)$ counting the number of unbordered factors is not 2-synchronized.

Proof. It is not hard to verify that $\mathbf{c}$ is 2-automatic and that $\mathbf{c}$ has exactly $r+2$ unbordered factors of length 2^r+1, for $r \geq 2$ — namely, the factors beginning at positions 2^i for $0 \leq i \leq r-1$, and the factors beginning at positions 2^{r+1} and $3 \cdot 2^r$. However, if $u_{\mathbf{c}}(n)$ were 2-synchronized, then reading an input where the first component looks like $0^i 1 0^{r-1} 1$ (and hence a representation of 2^r+1) for large r would force the transitions to enter a cycle. If the transitions in or before the cycle contained a nonzero entry in the second component, this would force $u_{\mathbf{c}}(n)$ to grow linearly with n when n is of the form 2^r+1. Otherwise, the corresponding transitions for the second component are just 0's, in which case $u_{\mathbf{c}}(n)$ is bounded above by a constant, for n of the form 2^r+1. Both cases lead to a contradiction. □