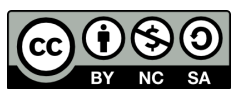
CDM

Decimation and Kernels

Klaus Sutner

Carnegie Mellon University

Fall 2025



We will now turn to the problem of testing automaticity, but using reverse base k instead of ordinary base k.

For the latter, we have a fairly elegant characterization in terms of morphisms and substitutions. Algorithmically, though, only pure fixed points without substitutions are easy to handle.

For reverse bases we will find a similarly elegant description, that produces a fairly straightforward algorithm. Unfortunately, it requires fairly long initial segements of the sequence.

1 **Decimation and Kernels**

2 **Uniform Morphisms**

3 **Periodic Sequences**

4 **State Complexity**

Recall that our definition of a k-automatic sequence is phrased in terms reverse base k, we need a CDFA $\mathcal{M}$ such that

$$A(n) = \lambda\Big(\delta(q_0, \mathsf{rrep}_k(n))\Big)$$

To find $\mathcal{M}$, or conclude that it fails to exist, we need an independent characterization of automaticity.

We will start with a very elegant way to describe p-automatic sequences when p is a prime, using algebra.

Then we present a more combinatorial approach that translates nicely into an algorithm.

Let $\mathbb{F} = \mathbb{F}_q$ be the finite field of characteristic p of size q.

Definition

The ring $\mathbb{F}[[X]]$ of formal power series over $\mathbb{F}$ has elements

$$f(X) = \sum_{n \geq 0} a_n X^n$$

where (a_n) is an infinite sequence over $\mathbb{F}$.

f is algebraic if there is a non-zero polynomial $P \in \mathbb{F}[X, Y]$ such that

$$P\big(X, f(X)\big) = 0$$

One can define addition and multiplication on $\mathbb{F}[[X]]$ in the usual way. We can think of $\mathbb{F}[X] \subseteq \mathbb{F}[[X]]$ as a subring.

Theorem

Let $f(X) = \sum_{n\geq 0} a_n X^n$ be a formal power series.

Then f is algebraic iff (a_n) is p-automatic.

For example, for the PTM sequence $T(X)$ over $\mathbb{F}_2$, we get

$$T(X) = X + X^2 + X^4 + X^7 + X^8 + X^{11} + X^{13} + X^{14} + X^{16} + X^{19} + \dots$$

$$0 = (X+1)^3 T(X)^2 + (X+1)^2 T(X) + X$$

For PTM this is easy to show by using the recurrence. Alas, in general, testing algebraicity is not so easy, we need a more flexible method.

$$
\begin{aligned}
T(X) &= \sum t_n X^n \\
&= \sum t_{2n} X^{2n} + \sum t_{2n+1} X^{2n+1} \\
&= \sum t_n X^{2n} + X \sum (t_n + 1) X^{2n} \\
&= T(X^2) + X\, T(X^2) + X \sum X^{2n} \\
&= (X+1) T(X^2) + \frac{X}{X^2+1}
\end{aligned}
$$

Exploiting characteristic 2 we get

$$
(X+1)^2\, T(X) = (X+1)^3\, T(X)^2 + X
$$

There is a general framework to study sequences of operations performed on some sort of objects. We are dealing with

- a collection X of objects,
- a collection S of operations on the carrier set X.

A priori, X is just a flat set with no particular structure, in particular there is no algebraic structure.

We can think of S as a collection of atomic actions that can be performed on the elements of X.

For example, X might be the collection of all squares in the plane, and our operations are rotations and reflections.

Or X might be the state set of a DFA, and the actions are given by the transition maps δ_a.

Performing a single operation is usually not particularly interesting, it is whole sequence of operations from S that produce interesting effects.

In the previous example, we might translate a rectangle, then rotate it, translate it again, and so on.

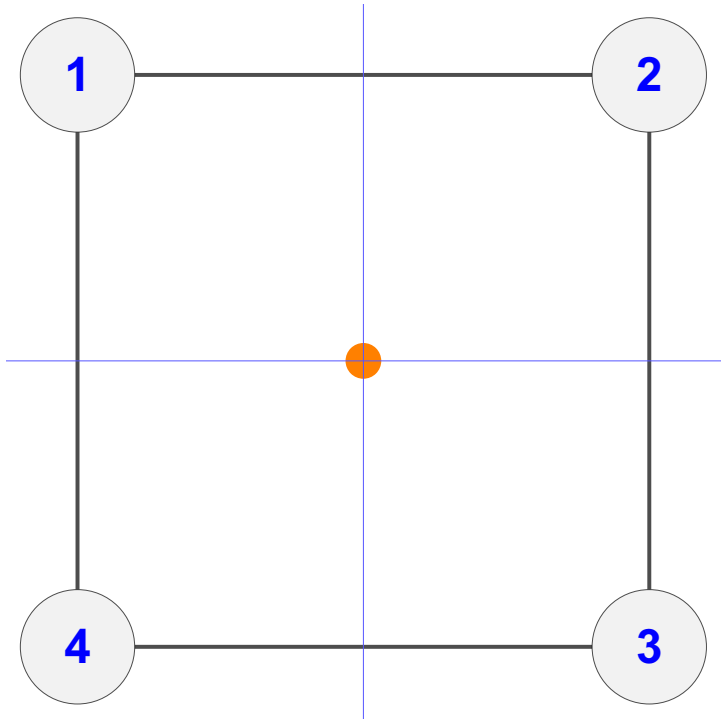
For the DFA, a sequence of atomic actions produces the extended transition function $\delta : \Sigma^\star \to Q \to Q$.

This is where algebra naturally pops up: sequences of atomic actions are naturally modeled by the free monoid $S^\star$, the collection of all finite sequences over S.

The monoid operation here is concatenating, which corresponds directly to executing one sequence of actions after another.

On the face of it, referring to the set of all finite sequences as the "free monoid over S" may sound a bit over the top.

But it opens the door to better descriptions. For example, let X be the collection of all placements of the unit square with labeled corners, centered at the origin.



Dihedral Groups

The square can be rotated around the center (say, clockwise by $\pi/2$) or reflected along the horizontal axis. Call these operations α and β.

It is easy to see that

$$\alpha^4 = 1$$
$$\beta^2 = 1$$
$$\alpha\beta = \beta\alpha^3$$

More work shows that these are essentially the only identities, so that the action sequences are really described by $\{\alpha, \beta\}^\star$/eqs.

But the last structure is (isomorphic to) the dihedral group D_4.

We want to build CPDFs for certain civilized sequences. Suppose we have some sequence $A \in \Delta^\omega$ and a CPDF $\mathcal{M}$ that generates it.

Burning Question: What is the meaning of the states in $\mathcal{M}$?

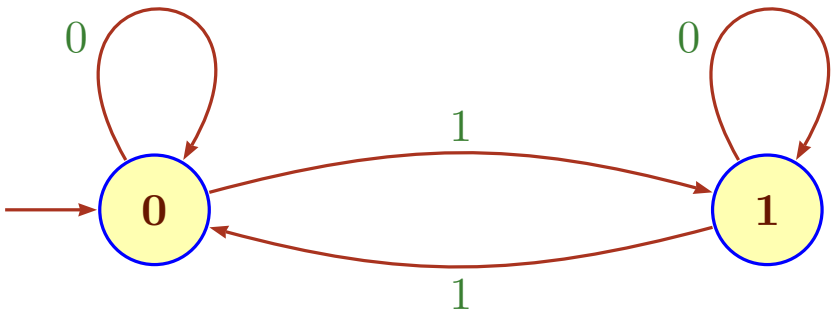
For example, what is the meaning of q_0 and $p = \delta(q_0, d)$?

Well, the behavior of q_0 better be A. Now recall that we write numbers in reverse base k, so d is the LSD in n. So $A(n) = A(d + kn')$ and we are only looking at some place in the subsequence $A(d + ik)_{i \in \mathbb{N}}$.

But then $p = \delta(q_0, d)$ should correspond to this subsequence.

And $\delta(p, d')$ should be a subsequence of this subsequence, and so on.

Example: PTM



$\delta(\mathbf{0}, 0)$ is the subsequence $T(2\mathbb{N}) = T$.

$\delta(\mathbf{0}, 1)$ is the subsequence $T(2\mathbb{N}+1) = \overline{T}$.

$\delta(\mathbf{1}, 0)$ is the subsequence $\overline{T}(2\mathbb{N}) = \overline{T}$.

$\delta(\mathbf{1}, 1)$ is the subsequence $\overline{T}(2\mathbb{N}+1) = T$.

Definition

Let $A \in \Delta^\omega$ be an infinite word. Given a stride $k \geq 1$ and an offset $d \geq 0$, define the decimation of A with respect to k and d by

$$A[k, d](i) = A(k \cdot i + d)$$

In other words, starting at position d, we only consider every kth letter. Usually this is most interesting when $0 \leq d < k$.

For the PTM word T we have

$$T[2, 0] = T \qquad T[2, 1] = \overline{T}$$
$$\overline{T}[2, 0] = \overline{T} \qquad \overline{T}[2, 1] = T$$

We can think of decimation as an action operating on Δ^ω. The atomic actions here are

$$A \rightsquigarrow A[k, d]$$

and we can compose these basic actions to get a decimation of a decimation of a decimation ...

Actions are arguably most useful when the underlying algebraic structure is a group (see the D_4 example above), but they are also useful in the more general setting of just semigroups.

Question: Is there some reasonable algebraic structure that describes our decimations?

Decimation Monoid

Proposition

The operation $*$ defined by

$$[k, d] * [k', d'] = [kk', kd' + d]$$

induces a monoid structure on $\mathbb{N}_+ \times \mathbb{N}$ with neutral element $[1, 0]$.

We refer to this monoid $\mathfrak{D}$ as the decimation monoid.

$\mathfrak{D}$ can be represent by matrices over $\mathbb{N}^{2\times 2}$ like so:

$$[k, d] \rightsquigarrow \left(\begin{smallmatrix} k & d \\ 0 & 1 \end{smallmatrix}\right)$$

The algebraically inclined will recognize $\mathfrak{D}$ as a semidirect product of the multiplicative monoid on $\mathbb{N}_+$ and the additive monoid on $\mathbb{N}$.

Denote the (left) shift operation by σ, so $\sigma(A)(i) = A(i+1)$. Clearly

$$\sigma(A)[k,d] = A[k,d+1]$$
$$\sigma\big(A[k,d]\big) = \sigma^k(A)[k,d]$$

The usual shuffle operation provides a sort of inverse to decimation since

$$A = \mathsf{shf}_{0\le d<k}\, A[k,d]$$

In other words, we take all the sequences obtained by stride k and offset $0 \le d < k$, and interleave them to get back the original sequence.

The decimations of the form $[k, d]$, $0 \leq d < k$ are called primary.
For us, the submonoid $\mathfrak{D}_k$ generated the primary decimations

$$[k, 0], [k, 1], \ldots, [k, k-1]$$

is particularly important.

Since $[k, d] * [k, d'] = [k^2, kd' + d]$ it is easy to see that

$$\mathfrak{D}_k = \{\, [k^i, d] \mid 1 \leq i, 0 \leq d < k^i \,\}$$

The submonoid is still infinite, but, if we apply its elements to a particular sequence A, we may well get only finitely many subsequences.

Let $0 \leq d < k^i$. Write fst(W) for the first letter of a word W. Then

$$A(d) = \text{fst}(A[k^i, d])$$

Thus, we can recover the letters of the word from the first letters of the various decimations.

Again, this is particularly interesting when the total number of these decimations is finite. This is the critical connection to finite state machines.

Definition

The k-kernel of a word $A \in \Delta^\omega$ is defined

$$\mathsf{Ker}_k(A) = \{\, A[k^i, d] \mid 1 \leq i, 0 \leq d < k^i \,\}$$

As we have seen, the 2-kernel of the PTM word T consists only of the two sequences T and $\overline{T} = T[2, 1]$.

Thus, the 2-automatic word T has a finite 2-kernel (don't get distracted by the fact that kernel actually has cardinality 2). Could this be coincidence?

Theorem

A sequence is k-automatic if, and only if, its k-kernel is finite.

Before we give a proof, consider a CPDF $\mathcal{M}$ generating a k-automatic sequence A.

Returning to our "what does a state mean" theme, let us interpret the behavior $[\![p]\!]$ of a state p in $\mathcal{M}$ to be the word generated with p as the initial state. Then we have

$$[\![\delta(p, d)]\!] = [\![p]\!][k, d]$$

This works since $\mathcal{M}$ is using reverse base k, so the first letter is the LSD. For standard base k one needs a different argument.

Correspondingly we can define an action of words over the digit alphabet $\mathcal{D}_k$ on Δ^ω by

$$[\![\delta(q_0, w)]\!] = A[k^{|w|}, \mathsf{rval}_k(w)]$$

So the states in $\mathcal{M}$ correspond to the decimations of A with strides k^i and offsets $j < k^i$. Recall that we can recover the letter in any particular position by filtering out the first letter in the decimation.

This is very similar in spirit to the characterization of the minimal DFA of a plain regular language as having states that correspond to the left quotients of the language.

Proof (of theorem)

First assume $A \in \Delta^\omega$ is k-automatic via a CPDF $\mathcal{M}$ on m states. As we have seen, every element in the k-kernel corresponds to a state in $\mathcal{M}$, so the kernel has cardinality at most m.

For the opposite direction consider the finite kernel K of A. Define a CPDF $\mathcal{M}$ by

$$\langle K, \mathcal{D}_k, \delta; A, \mathrm{fst} \rangle$$

where $\delta(X, d) = X[k, d]$.

□

1 Decimation and Kernels

2 Uniform Morphisms

3 Periodic Sequences

4 State Complexity

We will see in a moment how to explain automata using reverse base k representations, the question arises whether there is some natural combinatorial characterization of automaticity that is based on standard base k representations. Voilà.

Definition

A morphism $\mu : \Delta^\star \to \Delta^\star$ is k-uniform if $|\mu(a)| = k$ for all $a \in \Delta$.
μ is extensible if there is a special letter $a \in \Delta$ such that $\mu(a) = au$.

Note that every extensible morphism must have a fixed point in Δ^ω:

$$A = a\, u\, \mu(u)\, \mu^2(u)\, \mu^3(u) \ldots$$

If the morphism is also k-uniform, then a_n is mapped to the block $a_{kn}a_{kn+1} \ldots a_{kn+k-1} = \mu(a_n)$ under μ.

Here is a 2-uniform, extensible morphism for PTM:

$$0 \mapsto 01$$
$$1 \mapsto 10$$

which produces

$$0$$
$$01$$
$$0110$$
$$01101001$$
$$0110100110010110$$
$$01101001100101101001011001101001$$

Theorem

A sequence is k-automatic if, and only if, it is the image of a fixed point of a k-uniform morphism under an alphabetic substitution.

Proof.

Let $\mu : \Delta \to \Delta^k$ be a k-uniform morphism with fixed point B, $a \in \Delta$ the special letter, and $\sigma : \Delta \to \Delta$ a substitution.

We have to show that $A = \sigma(B)$ is k-automatic. Define a CDFA by using Δ as state set:

$$\mathcal{M} = \mathcal{M}(\mu, \sigma) = \langle \Delta, \mathcal{D}_k, \delta; a, \sigma \rangle$$

where $\delta(p, i) = \mu(p)_i$ (assuming 0-indexing).

An easy induction shows that $\lambda(\delta(a, \mathsf{rep}_k(n))) = A(n)$.

Proof, contd.

For the opposite direction assume we have a CDFA

$$\mathcal{M} = \langle Q, \mathcal{D}_k, \delta; q_0, \lambda \rangle$$

that generates a word $A \in \Delta^\omega$. We may safely assume that $\delta(q_0, 0) = q_0$.

We use Q as alphabet and define the morphism $\mu : Q \to Q^k$ by

$$\mu(p) = \delta(p, 0)\, \delta(p, 1) \ldots \delta(p, k-1) \in Q^k$$

The substitution is the coloring map λ.

Then $A = \lambda\big(\lim_{n\to\infty} \mu^n(q_0)\big)$.

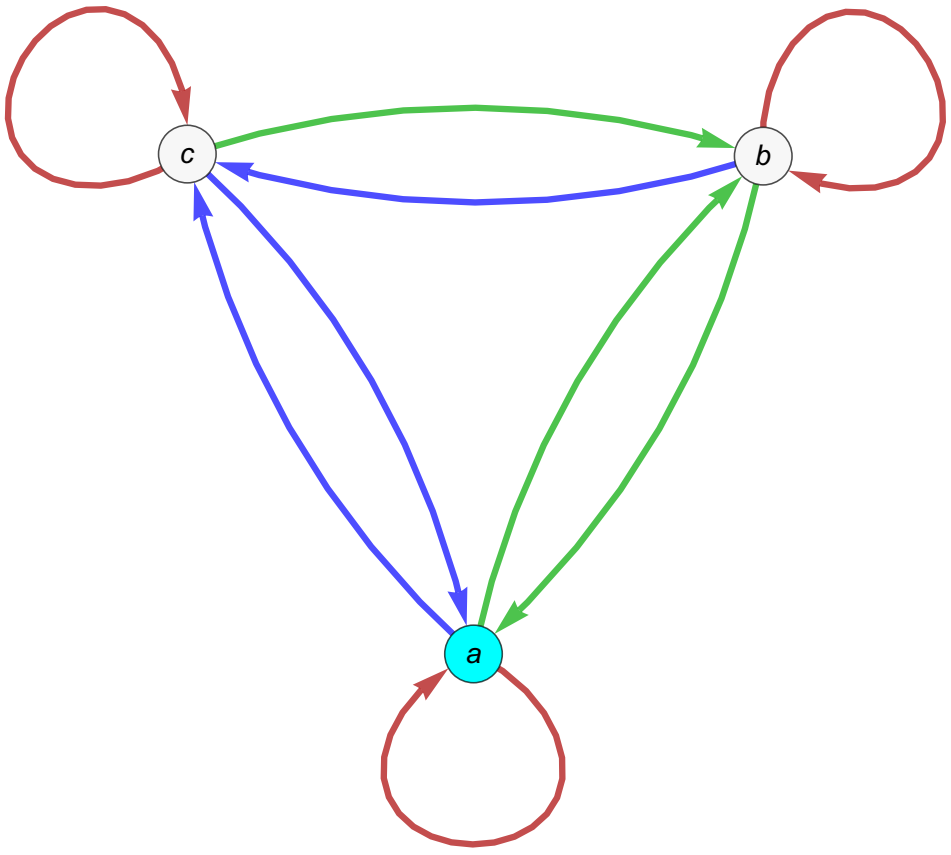
□

A 3-Uniform Morphism

$$a \mapsto abc \quad b \mapsto bac \quad c \mapsto cba$$

produces the fixed point

$$abc\ bac\ cba \quad bac\ abc\ cba \quad cba\ bac\ abc \quad bac\ abc\ cba \quad \ldots$$



$a \rightsquigarrow 0$ red
$b \rightsquigarrow 1$ green
$c \rightsquigarrow 2$ blue

And Inference?

Suppose we some infinite sequence A. E.g. we may have some way to compute $A(n)$, though we may only have an initial segment of A.

Either way, we may assume we know the alphabet Δ of A, $A \in \Delta^\omega$ and A uses all symbols in Δ.

> How would we check whether A is k-automatic?
> And construct a CDFA generating A if it is?

By the theorem, there has to some k-uniform morphism μ and an alphabetic substitution σ such that

$$A = \sigma\big(\mathsf{FPnt}(a, \mu)\big)$$
$$\mathsf{FPnt}(a, \mu) = a\, u\, \mu(u)\, \mu^2(u)\, \mu^3(u) \ldots$$

If we can determine μ and σ, we know how to construct a CDFA that generates A.

If no such μ and σ exist, A is not k-automatic.

Suppose we have additional information: the substitution σ is the identity, so A is a pure fixed point.

In this case we can read off μ from A:

$$A = a\ u_1 \ldots u_{k-1} \mid \mu(u_1) \mid \mu(u_2) \mid \ldots \mid \mu(u_{k-1}) \mid \ldots$$

Moreover, we only need an initial segment of A to obtain a complete description of μ.

If there is a substitution, the pure fixed point B is masked.

Still, we can resort to brute force and enumerate all substitutions $\sigma : \Delta \to \Delta$.

As stated, this is wildly exponential, though for small Δ it is still feasible.

Needless to say, one can limit the search space by skipping all substutions that clearly clash with the structure of the fixed point $B = \sigma^{-1}(A)$ (actually, there will be multiple preimages in general).

1 Decimation and Kernels

2 Uniform Morphisms

3 **Periodic Sequences**

4 State Complexity

Just to be clear, periodic sequences are not interesting as far as sequences inference is concerned (though there are lots of nice combinatorial problems).

They provide a nice we to test the k-automaticity machinery, though: we can directly compute the size of the k-kernel and thus of the minimal CDFA.

To see how decimation works, let's take a look at periodic sequences $A = \boldsymbol{a}^\omega$, where matters should be fairly straightforward. We call $\boldsymbol{a}$ the root of A.

Let $p = |\boldsymbol{a}|$, the period of the sequence. We use 0-indexing, since it causes less trouble with taking mods. Also, occasionally we write n^- for $n - 1$.

Write $\boldsymbol{a} = a_0, a_1, \ldots, a_{p^-}$, so that

$$A(n) = a_{n \bmod p}$$

From the definitions, we immediately have

$$A[k, d] = \text{take}\Big(A[k, d], p\Big)^\omega$$

where $\text{take}(X, l)$ indicates the first l elements of X.

Let $\boldsymbol{a} = a_0 a_1 \ldots a_p$- .

Decimations for stride $k = 2$. If p is even, we have

$$A[2,0] = (a_0 a_2 \ldots a_p\text{--})^\omega$$

$$A[2,1] = (a_1 a_3 \ldots a_p\text{-})^\omega$$

On the other hand, for odd p, we get

$$A[2,0] = (a_0 a_2 \ldots a_p\text{-} a_1 a_3 \ldots a_p\text{--})^\omega$$

$$A[2,1] = (a_1 a_3 \ldots a_p\text{--} a_0 a_2 \ldots a_p\text{-})^\omega$$

So in the even case, the period is cut in half, but we are not really concerned with the least period here. We could just double the word to get back to period p. From now on, for simplify we will assume that all the generators have length p.

Recall the decimation monoid $\mathcal{D} = \langle \mathbb{N}_+ \times \mathbb{N}, * \rangle$. $\mathcal{D}$ acts on all infinite sequences and in particular on periodic ones, but one might hope that things simplify a bit if we restrict ourselves to, say, p-periodic sequences.

First off, we don't need an infinite structure since for p-periodic A

$$A[k, d] = A[k \bmod p, d \bmod p]$$

For notational simplicity, we have written $A[0, d]$ for the constant sequence a_d^ω.

So we only need to consider the primary decimations for $0 \leq k < p$. Some finite algebraic decimation structure must be enough to handle p-periodic sequences.

Let's write $(n) = \{0, 1, \ldots, n^-\}$.

Definition

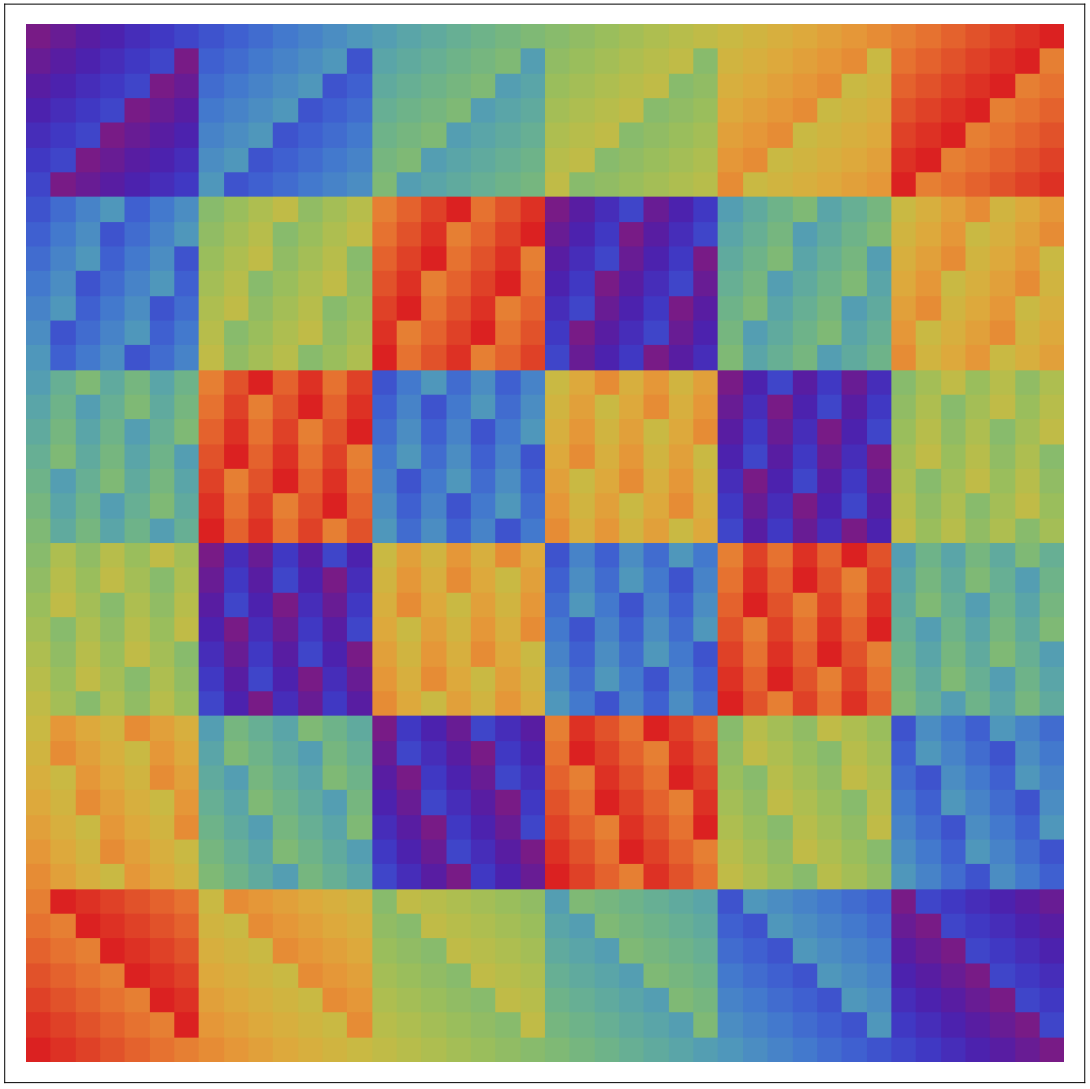
By a p-transformation we mean a map

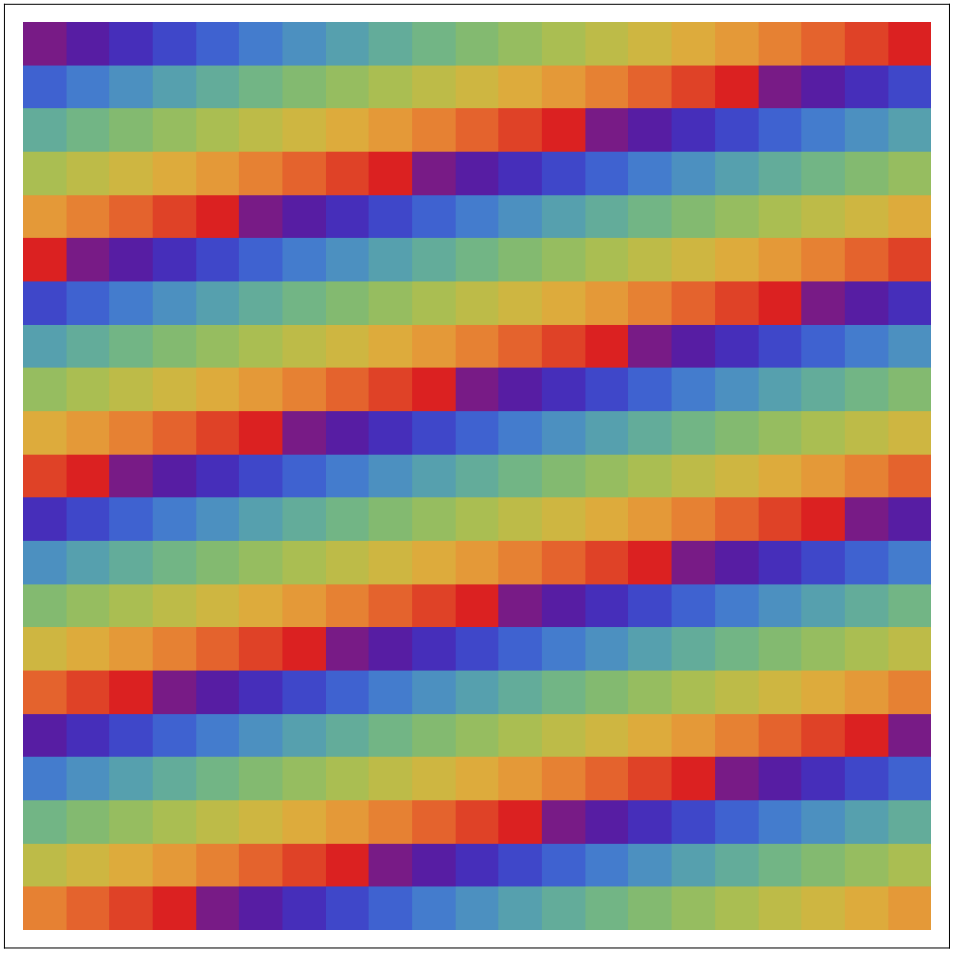
$$\begin{array}{rccc} T_{k,d}: & (p) & \longrightarrow & (p) \\ & s & \longmapsto & sk + d \bmod p \end{array}$$

where $1 \leq k < p$ and $0 \leq d < k$. The closure of all p-transformations under composition is the p-periodic transformation semigroup (PTS) $\mathcal{T}_p$.

If we restrict the transformations to a single stride k we obtain the p, k-periodic transformation semigroup $\mathcal{T}_{p,k}$.

In one-line notation, $T_{k,d}$ looks like $\big(d, k+d, 2k+d, \ldots, p^-k + d\big) \bmod p$.





p \ k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	2														
3	3	6													
4	4	8	8												
5	5	20	20	10											
6	6	12	6	6	12										
7	7	21	42	21	42	14									
8	8	24	16	16	16	24	16								
9	9	54	18	27	54	18	27	18							
10	10	40	40	20	10	10	40	40	20						
11	11	110	55	55	55	110	110	110	55	22					
12	12	36	24	12	24	24	24	24	12	24	24				
13	13	156	39	78	52	156	156	52	39	78	156	26			
14	14	42	84	42	84	28	14	14	42	84	42	84	28		
15	15	60	60	30	30	15	60	60	30	15	30	60	60	30	
16	16	64	64	32	64	64	32	32	32	64	64	32	64	64	32

To explain the table, write the transient/period of a geometric sequence modulo some fixed m as

$$\mathsf{tpgs}(a, m) = \text{ transient/period of the sequence } a^i \bmod m$$

E.g., $\mathsf{tpgs}(4, 17) = (0, 4)$ and $\mathsf{tpgs}(4, 18) = (1, 6)$.

Lemma

Let $(t, p) = \mathsf{tpgs}(a, m)$.
The order of $\mathcal{T}_{m,a}$ *is* $p\,m$ *whenever* $t = 0$, *and* $(t + p - 1)\,m$ *otherwise.*

Proof

The primary decimations all have stride k, so composition produces only decimations of the form

$$[k^i, d] \qquad 0 \leq d < k^i$$

Since A is p-periodic, we really are working with decimations

$$[k^i \bmod p, d \bmod p]$$

So the strides form a geometric sequence modulo p.

If k and p are coprime, the period of the sequence, times p, is the size of the monoid.

Otherwise the length of the whole lasso is $t + p - 1$, hence, this is the number of strides. Again multiply by p to account for the offsets.

□

If a and m are coprime, $t = 0$ and $p = \mathsf{ord}(a, \mathbb{Z}_m^*)$.

Otherwise let $p_1, \ldots, p_r$ be the primes in $\gcd(a, m)$. Then we can write the prime factorizations in the form

$$a = p_1^{e_1} \ldots p_r^{e_r} \alpha$$
$$m = p_1^{e'_1} \ldots p_r^{e'_r} \mu$$

where all the exponents are positive and p_i, α and μ are coprime. Then

$$t = \max_i \lceil e'_i / e_i \rceil$$
$$p = \mathsf{ord}(a, \mathbb{Z}_\mu^*)$$

are the transients and period of $(a^i \bmod m)$.

To see why, note that $\mathbb{Z}_m \simeq \mathbb{Z}_{m/\mu} \times \mathbb{Z}_\mu$.

1 Decimation and Kernels

2 Uniform Morphisms

3 Periodic Sequences

4 State Complexity

We can use the size of the minimal CDFA generating a k-automatic word as a measure of its complexity. An algorithm trying to identify a given sequence would typically try to build the minimal CDFA that generates it.

Definition

For any k-automatic sequence A we refer to the number of states of the minimal CDFA generating A as the state complexity of A.

Proposition

The state complexity of a k-automatic word is the size of its k-kernel.

Recall that we use reverse base k as the default representation.

As we have seen, there is also a minimal CDFA that generates A using standard base k. We refer to the state complexity of this machine as the state co-complexity of A.

On the face of it, complexity and co-complexity are exponentially related. Unfortunately, exponential blowup does happen.

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 $\text{rrep}_3(n) \in L_r$. Then the state complexity of A_r is $2^{r+2} + 1$, but the co-complexity is $r + 3$.

Suppose we are given a k-automatic word A. How can we compute its state complexity and the corresponding minimal CDFA?

Of course, this depends greatly on the representation of A.

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 CDFA using the standard closure algorithm: close $\{A\}$ under the operations $X \mapsto X[k, j]$, $0 \leq j < k$.

// generate k-kernel

let $K = \{A\}$
push A onto a stack S

while $S \neq$ nil
    let $X =$ pop(S)
    forall $j < k$ **do**
        $Y = X[k, j]$
        if $Y \in K$
        then record transition $X \xrightarrow{j} Y$
        else add Y to K, push onto S

return K

Again, there are only two non-logical operations in the algorithm:

- decimation, to compute $Y = X[k, j]$
- equality testing, to check if $Y \in K$

What if we start with a finite prefix $\alpha \sqsubset A$ rather than A itself?

Decimation extends to an operation over finite words in the obvious way. However, the length of the prefix shrinks 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 $\alpha =_p \beta$.

The Problem: Our our equality test may return false positives, there may well be two different kernel sequences whose initial segments we cannot distinguish.

As a consequence, out kernel algorithm will in general produce an under-approximation, we may obtain false identities.

The question then is: how long a prefix of A is required to properly separate all the kernel elements?

More precisely, how long does the prefix have to be in order to

- determine the size of the kernel,
- determine the minimal CDFA.

Theorem (KS, Tetruashvili)

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 CDFA.
Moreover, both bounds are tight.

Proof.

Suppose the k-kernel of A is $K = \{A, A_2, \ldots, A_m\}$. There are witnesses $v_i \in {\mathcal{D}_k}^\star$ 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 CDFA 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 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 CDFAs, yet they agree on their first $2^{2m-2}-1$ bits.

□