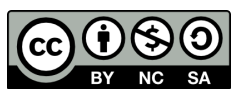
CDM

Automatic Sequences

Klaus Sutner

Carnegie Mellon University

Spring 2025



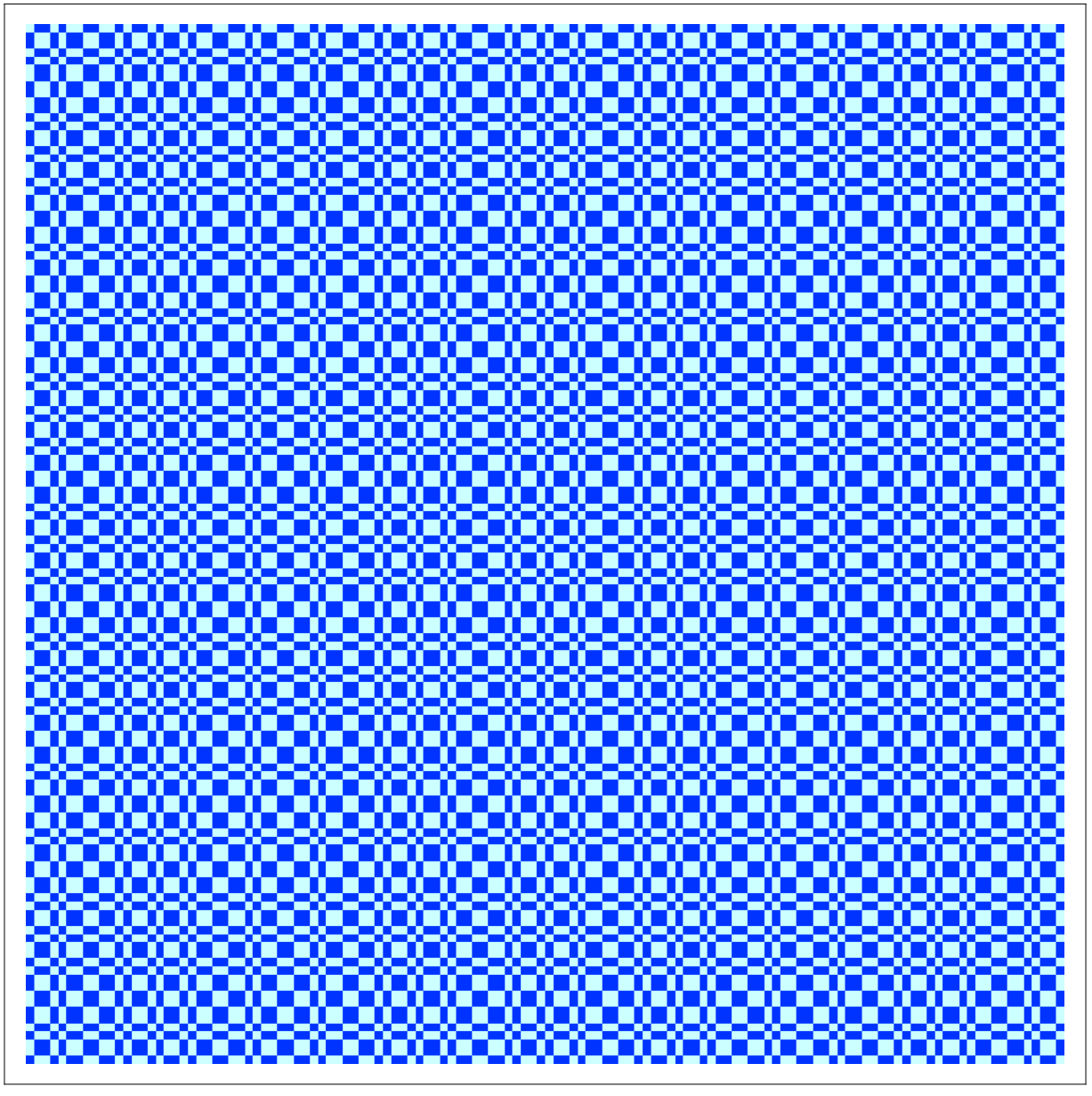
Here is a famous infinite binary word $T = (t_n)$ (discovered independently at least 3 times) defined by

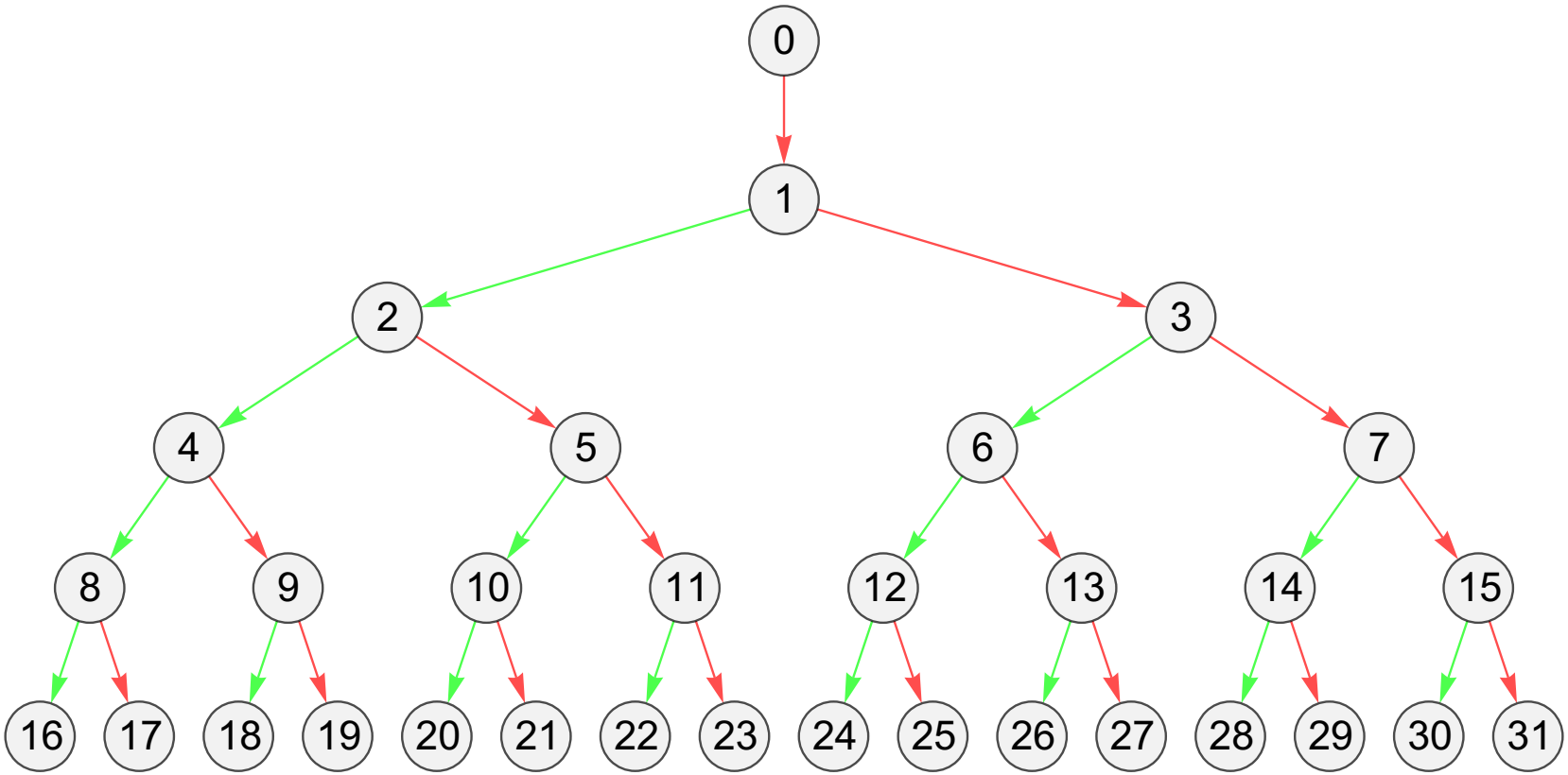
$$t_0 = 0$$
$$t_{2n} = t_n$$
$$t_{2n+1} = \overline{t_n}$$

Here $\overline{x}$ denotes bit-complement.

Let's write T_n for the binary word consisting of the first n bits of this sequence. For example,

$$T_{32} = 01101001100101101001011001101001$$




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
25
26
27
28
29
30
31

Recall the shuffle operation which interleaves the elements of two sequences:

$$a \parallel b = a_1 b_1 a_2 b_2 \ldots a_k b_k$$

Using shuffle, we can define a sequence of binary words as follows:

$$\begin{aligned} S_0 &= 0 \\ S_{n+1} &= S_n \parallel \overline{S_n} \end{aligned}$$

which produces

$$\begin{aligned} &01 \\ &0110 \\ &01101001 \\ &0110100110010110 \\ &01101001100101101001011001101001 \end{aligned}$$

Note the prefix property, so we can define $S = \lim_n S_n$.

Here is a kind of multiplication on binary words (known as Keane product):

$$x \otimes 0 = x \quad x \otimes 1 = \overline{x}$$
$$x \otimes yb = (x \otimes y)(x \otimes b)$$

For example, $01 \otimes 001 = 010110$. Clearly $|x \otimes B| = |x||B|$.

Now define

$$C_n = 0 \otimes \underbrace{01 \otimes 01 \otimes \ldots \otimes 01}_{n}$$

Again, C_n is a proper prefix of C_{n+1}, so we can define the limit $C = \lim C_n$.

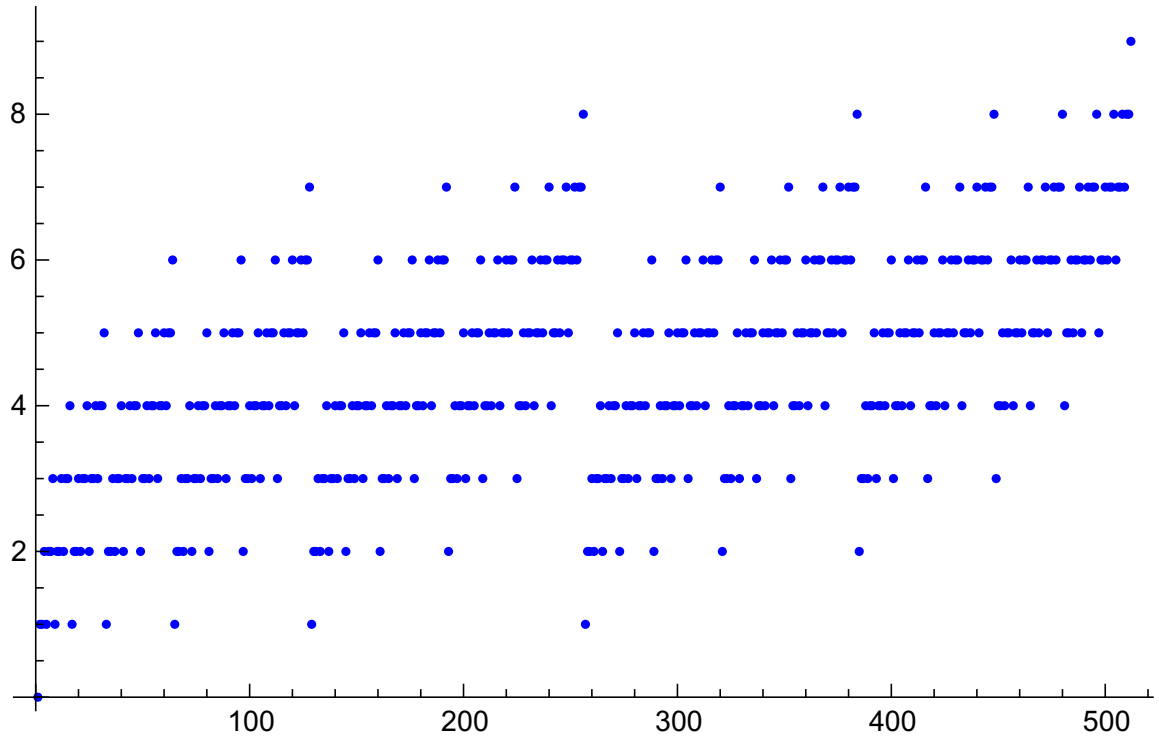
Define an infinite product over $\mathbb{Z}[x]$ as follows:

$$\prod_{k\geq 0}\left(1-x^{2^k}\right) = (1-x)(1-x^2)(1-x^4)\ldots$$
$$= \sum_{k\geq 0}(-1)^{p_k}x^k$$

where the coefficients $p_k \in \mathbf{2}$.

This produces yet another bit sequence $P = (p_k)$.

For simplicity, write $\sigma(n)$ for the binary digit sum of n, so $\sigma(10) = 2$ and $\sigma(255) = 8$ (of course, the argument is written in decimal). Define the digit sum sequence $D = \big(\sigma(n) \bmod 2\big)_n$ as the parities of all digit sums.



The digit sum function is quite erratic.

Lemma

They are all the same.

It is a bracing exercise to show that all the different definitions produce the exact same Prouhet-Thue-Morse sequence.

T has many interesting properties, perhaps the most important one being the fact that it is cube-free: it is impossible to write

$$T = \ldots x\, x\, x \ldots$$

for any non-empty finite word x. In fact, one cannot even get

$$T = \ldots x\, x\, x_1 \ldots$$

There is now a whole field, combinatorics of infinite words, where similar properties are studied. The arguments tend to be very combinatorial in nature (duh!) and often quite refractory.

Burnside Problem (1902)
Suppose G is a finitely generated group where all elements have finite order. Is G necessarily finite?

Some positive results were obtained initially, but the problem remained open for a good long time.

Theorem (Golod, Shafarevich 1964)

The Burnside Problem fails in general.

Bounded Version

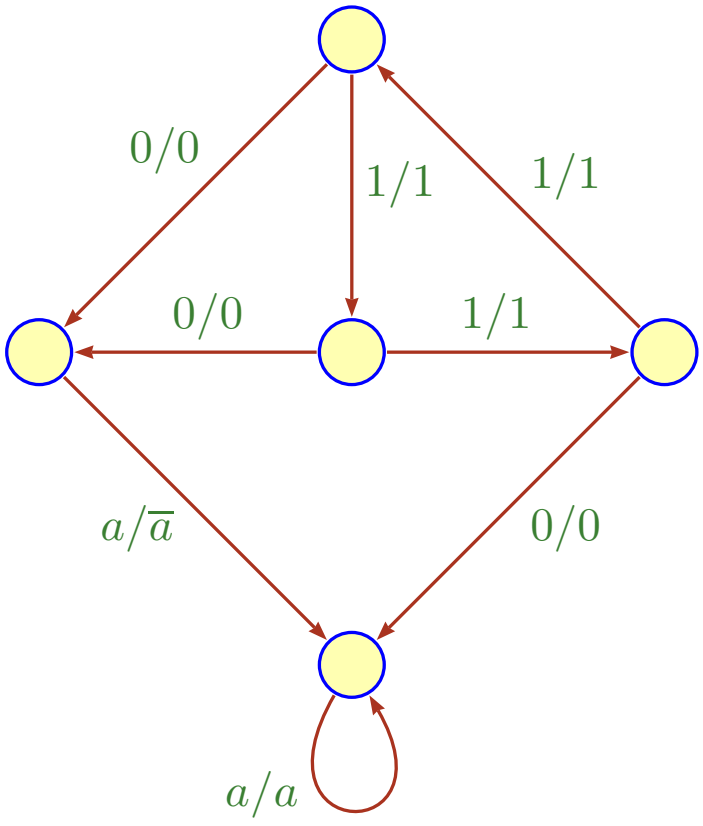
In the bounded version we require that all elements satisfy the identity $x^n = 1$ for some fixed n.

Theorem (Novikov, Adian 1968)

The bounded Burnside Problem fails for all odd $n > 4381$.

The PTM sequence plays a role in the (rather complicated) proof.

It is also directly applicable to the corresponding problem for semigroups.



This invertible Mealy automaton on 5 states due to Grigorchuk provides an example of a Burnside group (and has other interesting properties).

Think of an infinite word/sequence $A \in \Delta^\omega$ as a function $A : \mathbb{N} \to \Delta$. A word is computable if this function is computable: some Turing machine, on input n, computes $A(n)$.

Wild Idea:
How about words that can be computed by a finite state machine?

We need to modify the usual concept of a finite state machine to make this work: instead of scanning the word and accepting/rejecting, our machines will take $n \in \mathbb{N}$ as input and return a symbol in Δ.

There is a general class of finite state machines with output, so-called transducers. For the moment, we won't need this level of generality, we will simply attach an output symbol in Δ to each state in a DFA.

The other issue that requires clarification is the input $n \in \mathbb{N}$: we need to represent natural numbers as strings. We will only consider the two most pedestrian choices:

- ordinary base k representation
- reverse base k representation

The input alphabet will be a digit alphabet of the form

$$\mathcal{D}_k = \{0, 1, \ldots, k-1\}$$

Recall the standard base k (or radix k) representation of a natural number: the leading digit is the MSD:

$$\text{rep}_k(n) = d_r d_{r-1} \ldots d_0 \text{ where } n = \sum_{i \leq r} d_i k^i, d_r \neq 0$$

Given an arbitrary word w over $\mathcal{D}_k$, we will denote its value in base k by $\text{val}(w)$. It is convenient to allow ε for 0, so $\text{val}(0^i w) = \text{val}(w)$ for all $i \geq 0$, $w \in \mathcal{D}_k^*$.

We could change our conventions to obtain a unique notation system, but this approach is less cumbersome.

Alternatively, we can consider the leading digit to be the LSD, leading to reverse base k (or reverse radix k) notation:

$$\mathsf{rrep}_k(n) = d_r d_{r-1} \dots d_0 \text{ where } n = \sum_{i \le r} d_i k^{r-i}, d_r \neq 0$$

We write $\mathsf{rval}(w)$ for the numerical value of a word $w \in \mathcal{D}_k^*$.

Proposition

$\mathsf{rep}(n)^{\mathsf{op}} = \mathsf{rrep}(n)$

Since regular languages are closed under reversal one may suspect that both notation systems will produce the same results (but see the section on state complexity below).

Since we want to determine $A(n) \in \Delta$ given (the representation of) n we need to modify ordinary DFAs slightly to produce output. Think of the symbols in Δ as colors.

Definition

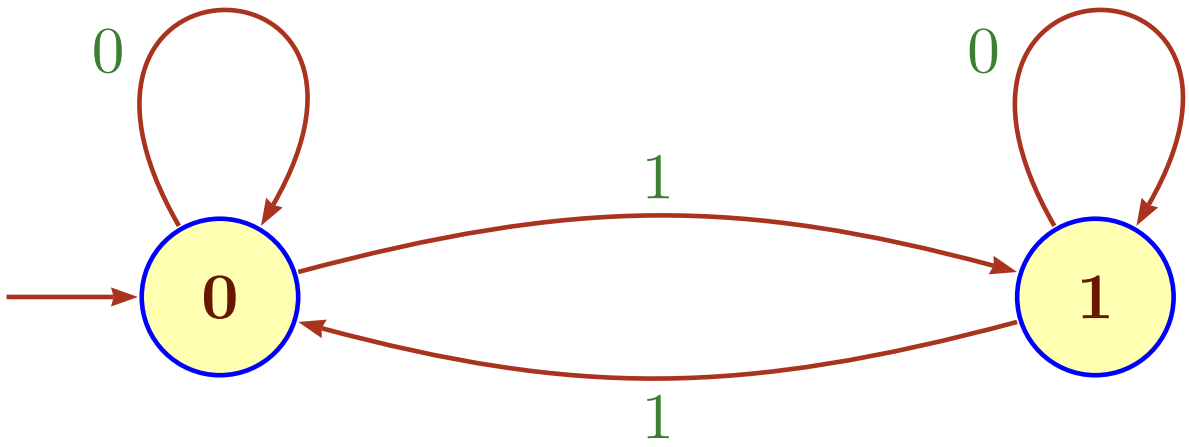
A colored deterministic finite automaton (CDFA) is an automaton

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

where $\langle Q, \Sigma, \delta \rangle$ is a deterministic and complete transition system, $q_0 \in Q$ an initial state and $\lambda : Q \to \Delta$ is the coloring map.
The cardinality of Δ is the order of $\mathcal{M}$.

Given input $\mathsf{rep}(n) \in \mathcal{D}^\star$, we define the output to be

$$\lambda\big(\delta(q_0, \mathsf{rep}(n))\big)$$



In this case, $\mathcal{D} = \Delta = \mathbf{2}$.

Of all the alternative definitions of the PTM sequence, the digit sum one is arguably the most basic.

One can check that it does not matter whether we use base 2 or reverse base 2.

A CDFA of order 2 is essentially just a plain DFA, so we should expect the standard machinery for DFAs to carry over.

For $c \in \Delta$ the c-behavior of a state p is

$$[\![p]\!]_c = \{\, x \in \mathcal{D}^\star \mid \lambda(\delta(p, x)) = c \,\}$$

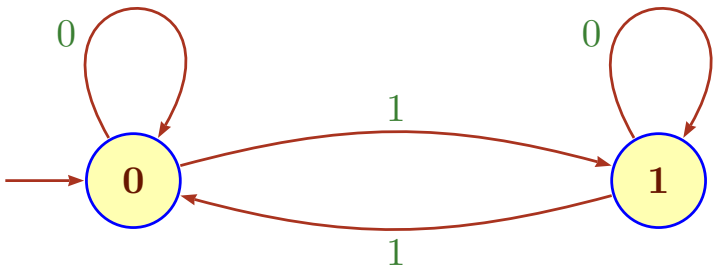
The behavior of p is the vector $([\![p]\!]_c \;\big|\; c \in \Delta)$.

In slight abuse of terminology, we define the language of $\mathcal{M}$ to be the vector of languages

$$\mathcal{L}(\mathcal{M}) = [\![q_0]\!] \subseteq (\mathcal{D}^\star)^{|\Delta|}$$

A CDFA is accessible if $\forall\, p\, \exists\, x\, \big(\delta(q_0, x) = p\big)$; it is reduced if its states all have distinct behavior.

The PTM machine



has language

$$\big(\{\, x \in \mathbf{2}^\star \mid \#_1 x \bmod 2 = 0 \,\}, \{\, x \in \mathbf{2}^\star \mid \#_1 x \bmod 2 = 1 \,\}\big)$$

and is clearly accessible and reduced.

As with ordinary DFAs we are interested in the smallest equivalent (accepting the same language) CDFA.

Theorem

For every CDFA there is an equivalent one that is accessible and reduced. Moreover, this CDFA is unique up to isomorphism.

We will refer to this automaton as the minimal CDFA.

The minimal CDFA can be computed using essentially the same algorithms as in the order 2 case.

Start with the partition of Q induced by the coloring λ.

Then keep breaking blocks in the partition apart until we reach a point where $P = (P_1, \ldots, P_m)$ and for all $d \in \mathcal{D}$:

$$\delta(P_i, d) \subseteq P_j \quad \text{or} \quad \delta(P_i, d) \cap P_j = \emptyset$$

In this case we can think of P_i as mapping to P_j under d and get a potentially smaller CDFA.

When a fixed point is reached, we have the minimal automaton. If n is the state complexity of $\mathcal{A}$, this is easy to do in $O(n^2)$ steps. It can even be handled in $O(n \log n)$ steps, but the algorithms are more complicated.

Definition

An infinite word A over alphabet Δ is k-automatic if there exists a CDFA $\mathcal{M}$ over alphabets $\mathcal{D} = \mathcal{D}_k$ and Δ such that for all $w \in \mathcal{D}^\star$:

$$\lambda(\delta(q_0, w)) = A(\mathsf{rval}_k(w))$$

We will say that $\mathcal{M}$ generates A to distinguish this situation from other forms of acceptance of infinite words.

Thus, our definition assumes reverse base k representations. This is somewhat arbitrary, but will turn out to be convenient later.

But Why?

The definition is actually a bit strange, one might expect more parameters, something like

reverse-k-trailzeros-automatic or
k-noleadzeros-noempty-automatic

This is not only ugly, it also could ruin the whole idea: if these different versions all turn out to be different, the definitions are too brittle to be of any interest.

So we have to make sure that all the variants are "the same" in the sense that a sequence is either automatic in all of them or in none.

To do this, we need to explain exactly what kind of notation systems for numbers we want to allow for automaticity, and then establish equivalence.

Theorem

Let A be an infinite word. Then A is k-automatic iff one of the following conditions hold: A is generated by a CDFA

- *using base k notation (with or w/o leading zeros)*
- *using reverse base k notation (with or w/o trailing zeros)*

Leading/trailing zeros and the question whether the empty word denote 0 are easy to deal with, the interesting problem is the relation between standard base k and reverse base k.

This is analogous to plain regular languages being closed under reversal.

There are several standard minimization algorithms: Moore, Hopcroft, Valmari; all based on partition refinement. But there is also a positively weird one, due to Brzozowski. Instead of refining partitions, it uses reversal and Rabin-Scott determinization to construct the minimal automaton—which operations, at first glance, seem to have absolutely nothing to do with the task at hand.

Write

- $\mathcal{A}^{\mathrm{op}}$ for the reversal of any finite state machine, and
- $\mathrm{pow}(\mathcal{A})$ for the accessible part obtained by determinization.

Determinization preserves the language, but may be exponential in time and space. Reversal of the automaton reverses the language, but is linear time.

Lemma

If $\mathcal{A}$ is an accessible DFA, then $\mathcal{A}' = \mathsf{pow}(\mathcal{A}^{\mathsf{op}})$ is minimal.

Proof.

Let $\mathcal{A} = \langle Q, \Sigma, \delta; q_0, F \rangle$.

$\mathcal{A}'$ is accessible by construction, so we only need to show that any two states have different behavior.

Let $P = \delta_x^{-1}(F) \neq P' = \delta_y^{-1}(F)$ in $\mathcal{A}'$ for some $x, y \in \Sigma^\star$.

We may safely assume that $p \in P - P'$.

Since $\mathcal{A}$ is accessible, there is a word z such that $p = \delta_z(q_0)$.

Since $\mathcal{A}$ is deterministic, z^{op} is in the $\mathcal{A}'$-behavior of P but not of P'.

□

More generally, we can use the lemma to establish the following surprising minimization algorithm.

Theorem (Brzozowski 1963)

Let $\mathcal{A}$ be a finite state machine. Then the automaton $\mathsf{pow}(\mathsf{pow}(\mathcal{A}^{\mathsf{op}})^{\mathsf{op}})$ *is (isomorphic to) the minimal automaton of $\mathcal{A}$.*

Proof.

$\mathcal{B} = \mathsf{pow}(\mathcal{A}^{\mathsf{op}})$ is an accessible DFA accepting $\mathcal{L}(\mathcal{A})^{\mathsf{op}}$ for any finite state machine $\mathcal{A}$ (deterministic or not).

By the lemma, $\mathcal{A}' = \mathsf{pow}(\mathcal{B}^{\mathsf{op}})$ is the minimal automaton accepting $\mathcal{L}(\mathcal{A})^{\mathsf{op}\,\mathsf{op}} = \mathcal{L}(\mathcal{A})$.

□

Dire Warning

We are only ever constructing the accessible part of the full power automaton. Alas, in general, that accessible part already can have exponential size.

So Brzozowski's algorithm can be exponential, unlike the "real" algorithms.

Weird Fact: It often works quite well, when the machines involved are small for whatever reason, the two constituent algorithms are quite fast.

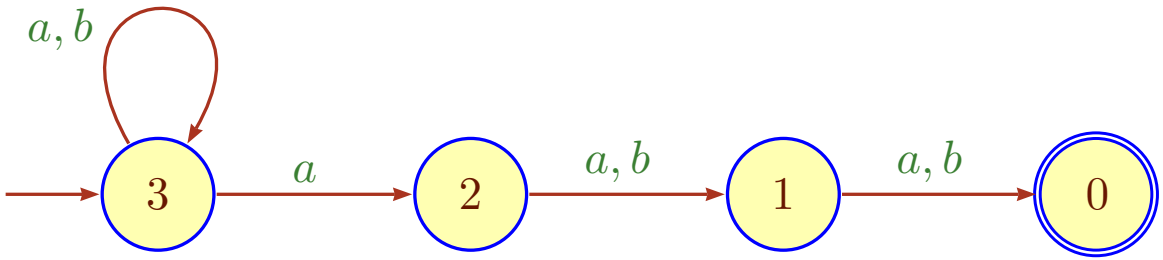
On occasion the last lemma can be used to determine minimal automata abstractly, without actually running any algorithm.

> Structural descriptions are better than algorithmic ones.

More precisely, in some cases we can make assertions about the minimality of a collection of finite state machines using Brzozowski's result that would be more difficult to obtain in other ways. Running a minimization algorithm abstractly on a class of machines is usually quite hard (though there are a few exceptions like tally DFAs). A characterization like Brzozowski's is usually better to work with.

Example: kth-Letter

Recall the standard NFA $\mathcal{A}$ for "kth symbol from the end."



Claim: pow($\mathcal{A}$) has 2^k states and is minimal.

Since $\mathcal{A} = \mathcal{B}^{\text{op}}$ where $\mathcal{B}$ is the minimal partial DFA for "kth symbol from the front," we have pow($\mathcal{A}$) = pow($\mathcal{B}^{\text{op}}$), done by Brzozowski's lemma.

State complexity 2^k is purely combinatorial: exactly the states $P \subseteq [0, k]$ such that $k \in P$ belong to pow($\mathcal{A}$).

This is borderline embarrassing, but a cheap source of automatic sequences is to start with a regular language $L \subseteq \mathcal{D}_k^*$ that is closed under trailing zeros, $L0^\star = L$.

We can associate a binary sequence $A_L \in \mathbf{2}^\omega$ with L by

$$A_L(n) = \begin{cases} 1 & \text{if } \mathsf{rrep}_k(n) \in L, \\ 0 & \text{otherwise.} \end{cases}$$

A_L is k-automatic by brute-force. In fact, the minimal CDFA for A_L is really just the minimal DFA for L.

We can do the same with ordinary base k representation, in which case we need to deal with leading zeros: $0^\star L = L$.

Fix some modulus $m \geq 2$ and base k; define the divisibility language

$$L = \{\, x \in \mathcal{D}_k^* \mid \mathsf{val}_k(x) = 0 \pmod{m} \,\}$$

The corresponding sequence $A_L \in \mathbf{2}^\omega$ is particularly simple

$$A_L = (\, 1\, 0^{m-1} \,)^\omega$$

Surprisingly, it's already far from obvious to figure out what the corresponding minimal CDFA looks like.

Since we are interested in values modulo m, it is natural to choose the modular numbers $Q = \{0, 1, \ldots, m-1\}$ as state set, $q_0 = 0$, $\lambda(0) = 1$, $\lambda(p) = 0$ otherwise. Again, this is really a standard DFA acceptor over digit alphabet $\mathcal{D}_k$ in disguise, with $F = \{0\}$.

For the transition function note that

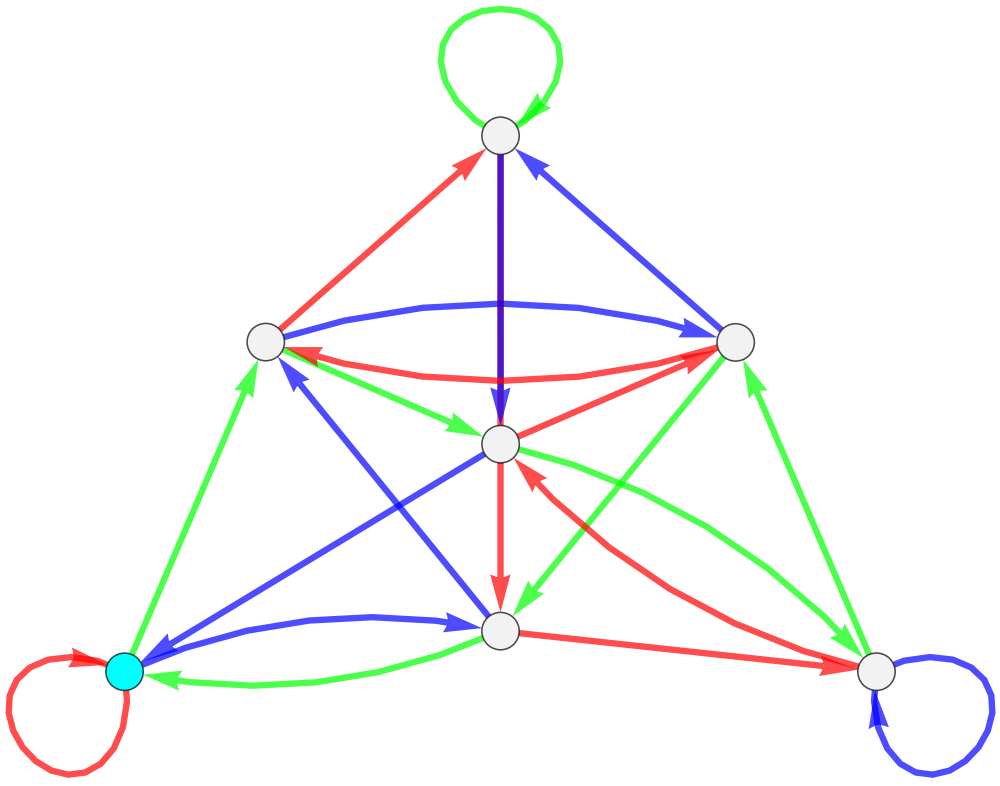
$$\mathsf{val}(xd) = \Big(k\,\mathsf{val}(x) + d\Big) \bmod m$$

so we can set

$$\delta(p, d) = \Big(p\,k + d\Big) \bmod m$$

We will call these machines Horner automata, in symbols $\mathcal{H}_{m,k}$.

Careful, though, $\mathcal{H}_{m,k}$ is not minimal in general, just think about $\mathcal{H}_{m,m}$.
It's the obvious machine that works, but it typically has redundant states.



$\mathcal{H}_{7,3}$; edge colors: red–0, green–1, blue–2.

This is more complicated since

$$\mathsf{rval}(xd) = \left(\mathsf{rval}(x) + d\,k^{|x|}\right) \bmod m$$

so the machine needs to keep track of the numbers $k^{|x|} \bmod m$.

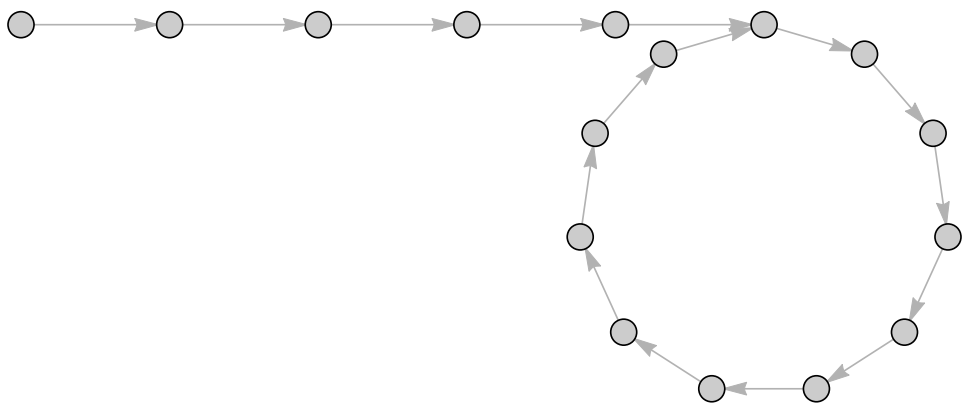
The sequence $k^i \bmod m$, $i \geq 0$, depends very much on divisibility properties of m and k. Without worrying about the details, it has to be ultimately periodic as in $\alpha\,\beta^\omega$ where $|\alpha| + |\beta| \leq m$, where α and β are finite words over Σ_m. We call α, β the fundamental sequence of k and m.

We can wrap this up in a nice algebraic way.

Here is a function that describes the position of a particle moving on a lasso with transient t and period p.

$$\mathsf{rem}_{t,p}(i) = \begin{cases} i & \text{if } i < t, \\ t + (i - t) \bmod p & \text{otherwise.} \end{cases}$$

Write $\mathsf{succ}(i) = \mathsf{rem}_{t,p}(i+1)$ and write $\mathbb{N}_{t,p}$ for the structure[†] $\langle\, \{0, 1, \ldots, t{+}p{-}1\}, \mathsf{succ} \,\rangle$.



A picture of $\mathbb{N}_{5,11}$.

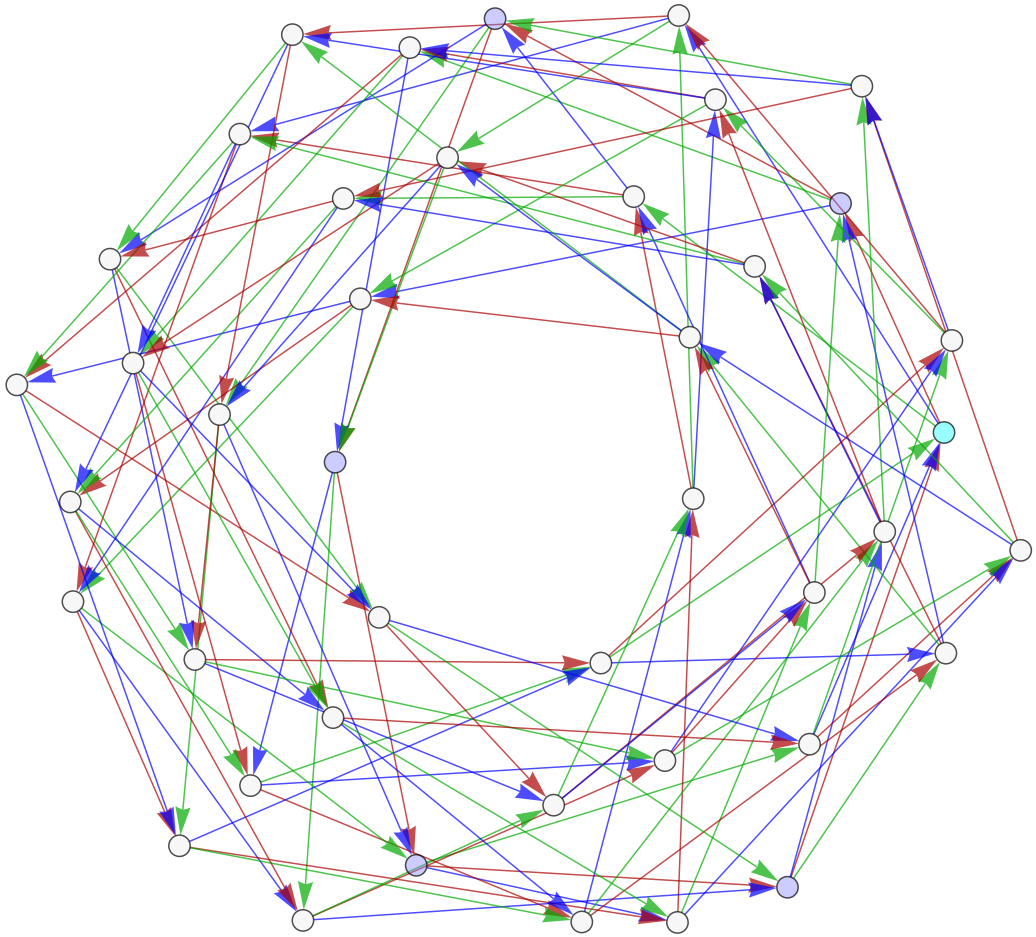
[†]Sometimes called a semimodule, though there is really very little algebraic structure.

To build the divisibility machine for reverse base k, let t and p be the transient/period of the sequence $k^i \bmod m$, $i \geq 0$ and let α, β be the fundamental sequence of k and m.

We can now build a divisibility automaton $\mathcal{D}_{m,k}$ as follows:

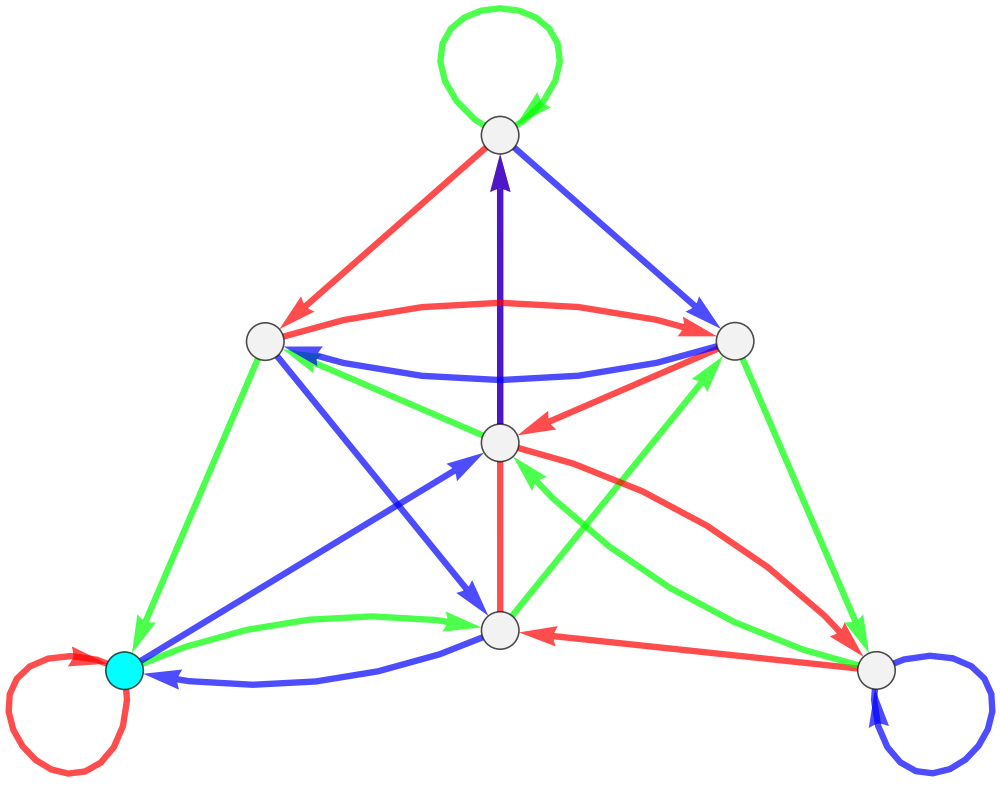
$$\begin{aligned} Q &= \{0, \ldots, m-1\} \times \mathbb{N}_{t,p} \\ \delta((p,s),d) &= \big(p + d\,\beta_s \bmod m, \mathsf{succ}(s)\big) \\ q_0 &= (0,0) \\ F &= \{\, (0,s) \mid s \in \mathbb{N}_{t,p} \,\} \end{aligned}$$

Note that this machine has size $O(m^2)$, though the accessible part can be smaller and the minimal machine can be smaller yet.

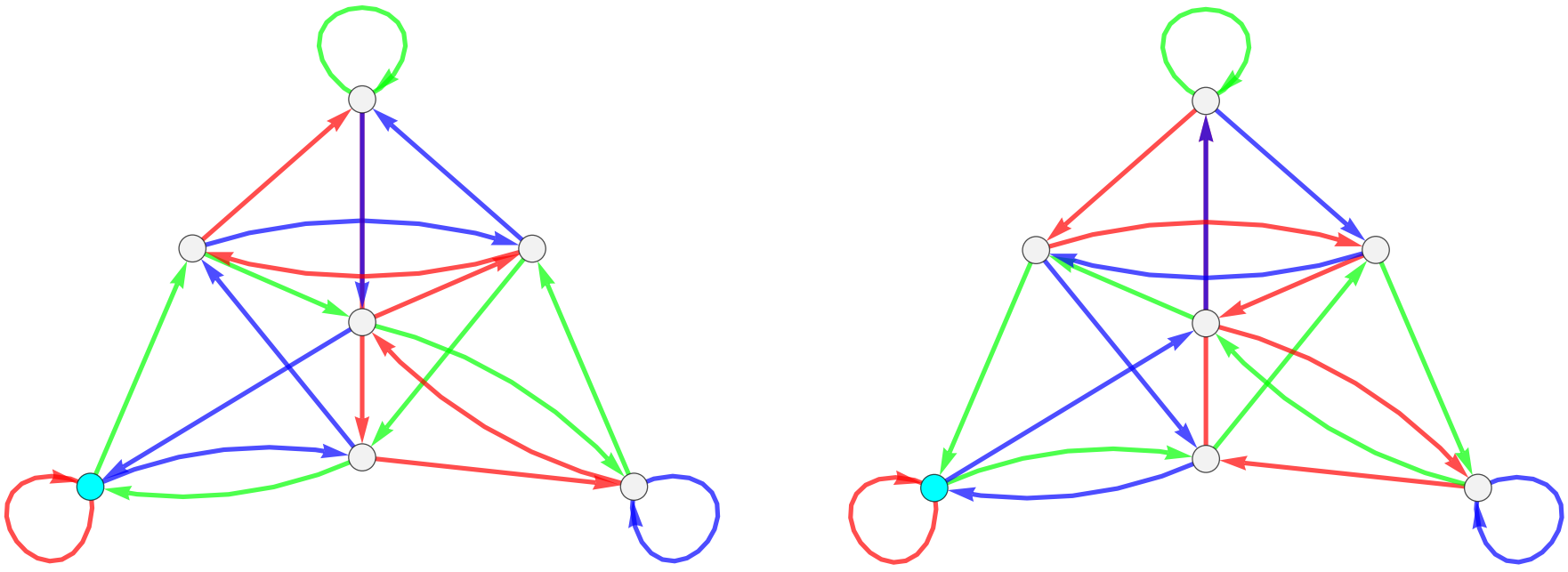


$\mathcal{D}_{7,3}$ has $42 = 7 \cdot 6$ states.

Fundamental sequence is $\alpha =$ nil, $\beta = 1, 3, 2, 6, 4, 5$ (transient 0, period 6).



The minimal automaton of $\mathcal{D}_{7,3}$.



Plain base 3 on the left, reverse base 3 on the right.

But Why?

From the picture, the minimal automaton of $\mathcal{D}_{7,3}$ is the reversal of $\mathcal{H}_{7,3}$.

To see why, note that $\mathcal{H}_{7,3}$ is a permutation automaton: all the transition functions δ_a are permutations of the state set.

By Brzozowski, that means that $\mathrm{pow}\left(\mathcal{H}_{7,3}^{\mathrm{op}}\right)$ is the minimal automaton for reverse base 3. But $\mathcal{H}_{7,3}^{\mathrm{op}}$ is a DFA, done.

Careful, though, it is not true in general that $\mathcal{H}_{m,k}$ is a permutation automaton.

Let's get back to our real issue: we would like there to be no difference between automaticity defined by base k or defined by reverse base k.

Since $\text{rrep}_k(n) = \text{rep}_k(n)^{\text{op}}$, we might get some inspiration from Brzozowski's wild minimization method. Careful, though: as stated, Brzozowski's result holds only for order 2 automata (final vs non-final states), so we need tweak things a little bit to handle colors in general.

More precisely, suppose $\mathcal{M}$ is the minimal CDFA for reverse base k that generates some sequence $A \in \Delta^\omega$. We need to construct the minimal CDFA $\mathcal{M}'$ that generates A using ordinary base k.

Consider the original minimal CDFA $\mathcal{M}$ for A in reverse base k:

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

Define Δ-many accessible ordinary DFAs of the form

$$\mathcal{M}_a = \langle Q, \Sigma, \delta; q_0, \lambda^{-1}(a) \rangle \qquad \text{for } a \in \Delta$$

$$\mathcal{M}'_a = \mathsf{pow}\left(\mathcal{M}_a^{\mathsf{op}}\right)$$

Form a product automaton from all these machines

$$\mathcal{M}' = \bigotimes_{a \in \Delta} \mathcal{M}'_a$$

The states of $\mathcal{M}'$ are Δ-vectors of subsets of Q:

$$\boldsymbol{p} = (p_a, p_b, \ldots, p_c) \in \mathfrak{P}(Q)^{\Delta}$$

The transition function of $\mathcal{M}'$ applies the inverse δ^{op} of the old δ to all the components of the state vector.

We turn the DFA $\mathcal{M}'$ into a CDFA via the coloring

$$\lambda(\boldsymbol{p}) = a \quad \text{iff} \quad q_0 \in p_a$$

We will see in a moment that that this is well-defined.

Theorem

The CDFA $\mathcal{M}'$ is minimal and isomorphic to $\mathcal{M}^{\text{op}}$.

Proof. First note that by construction $\mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M})^{\text{op}}$.

A simple induction shows that each state vector $\boldsymbol{p}$ of $\mathcal{M}'$ forms a partition of the original state set Q, so suppose that $\boldsymbol{p} \neq \boldsymbol{q}$, say, $p_a \neq q_a$. We may safely assume that $p \in p_a - q_a$.

But $\mathcal{M}$ is accessible, so $\delta(q_0, x) = p$ for some word x. It follows that x^{op} lies in the a-behavior of $\boldsymbol{p}$ but not of $\boldsymbol{q}$.

Hence $\mathcal{M}'$ is both accessible and reduced. Done.

□

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

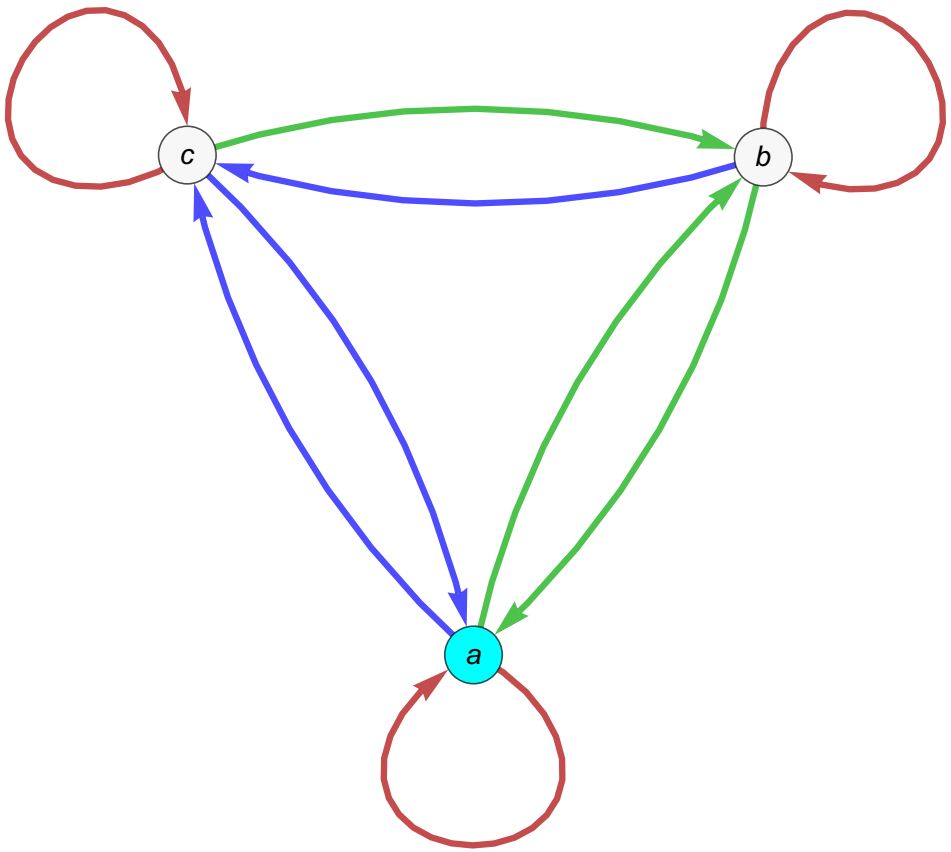
□

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

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).