The Mathematics of Iterated Morphisms

Jeffrey Shallit
Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@graceland.uwaterloo.ca
http://www.math.uwaterloo.ca/~shallit

Morphisms

Let Σ be a finite alphabet.

By Σ^* we mean the set of all finite strings over Σ.

For example,

$$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}.$$

Here ϵ denotes the *empty string*, the unique string of length 0.

A *morphism* is a map $h : \Sigma^* \to \Delta^*$ such that

$$h(xy) = h(x)h(y)$$

for all $x, y \in \Sigma^*$.

To define a morphism h, it suffices to specify the image $h(a)$ for each $a \in \Sigma$.

Some Examples

Define the *Thue-Morse* morphism μ by

$$\begin{aligned}\mu(0) &= 01\\ \mu(1) &= 10.\end{aligned}$$

We studied this morphism in a previous talk.

Here's another example. Let $\Sigma = \{a_r, a_l, b_r, b_l\}$. Define

$$\begin{aligned}\varphi(a_r) &= a_l b_r\\ \varphi(a_l) &= b_l a_r\\ \varphi(b_r) &= a_r\\ \varphi(b_l) &= a_l\end{aligned}$$

Then applying φ models the growth of an organism, the blue-green bacterium *Anabaena catenula*. Here a_l, a_r denote "fat" cells, and b_l, b_r denote "thin" cells.

Growth of Anabaena catenula

$$\begin{aligned}\varphi(a_r) &= a_l b_r \\ \varphi(a_l) &= b_l a_r \\ \varphi(b_r) &= a_r \\ \varphi(b_l) &= a_l\end{aligned}$$

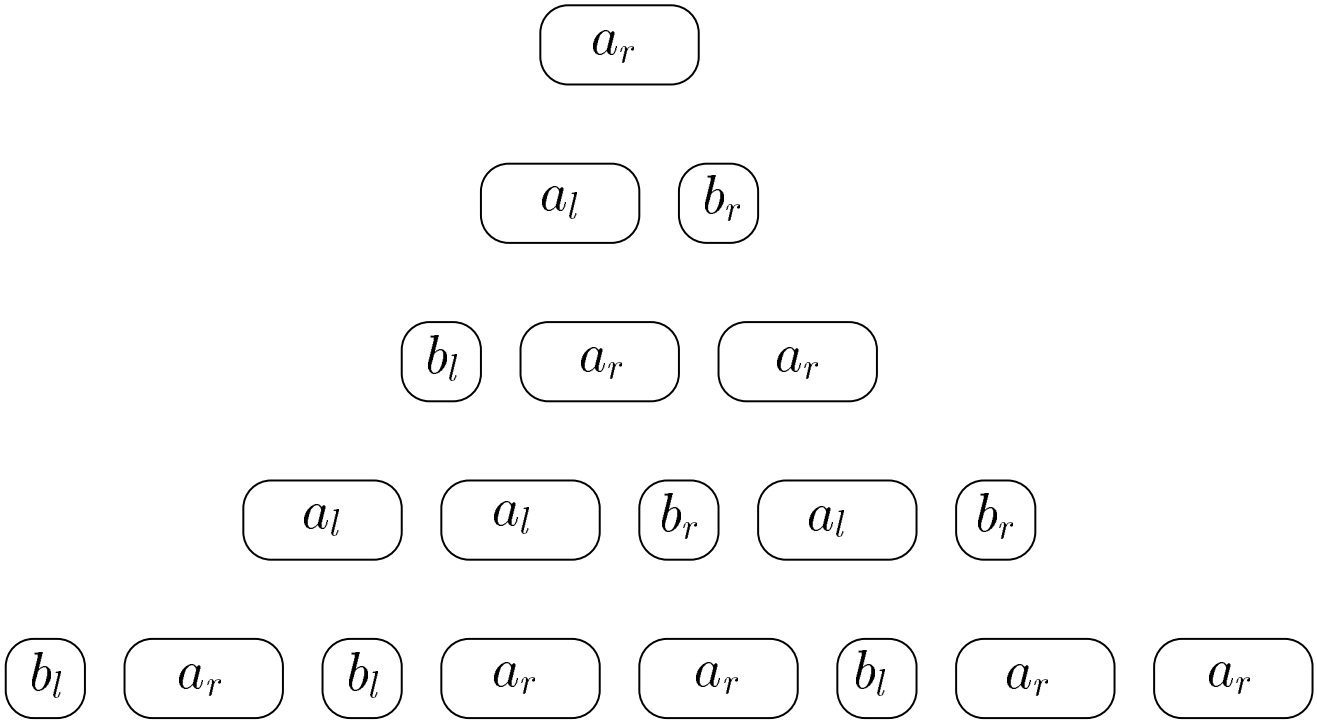


Figure 1: Growth of *Anabaena catenula*

Iterating Morphisms

If $h : \Sigma^* \to \Sigma^*$, then we can iterate h.

We write

$$\begin{aligned} h^0(x) &= x \\ h^1(x) &= h(x) \\ h^2(x) &= h(h(x)) \\ h^3(x) &= h(h(h(x))) \\ &\vdots \end{aligned}$$

Iterating μ gives, in the limit, the Thue-Morse infinite word

$$\mathbf{t} = 0110100110010110\cdots$$

which I discussed in a previous talk.

In general, if there is a letter a such that $h(a) = ax$, then we can define

$$h^\omega(a) = a\ x\ h(x)\ h^2(x)\ h^3(x)\cdots,$$

a *fixed point* of h starting with a.

Fixed Points of Morphisms

A *fixed point* of a map h is an element of its domain x such that $x = h(x)$.

To see that

$$h^\omega(a) = a\ x\ h(x)\ h^2(x)\ h^3(x) \cdots$$

is a fixed point of h, we calculate

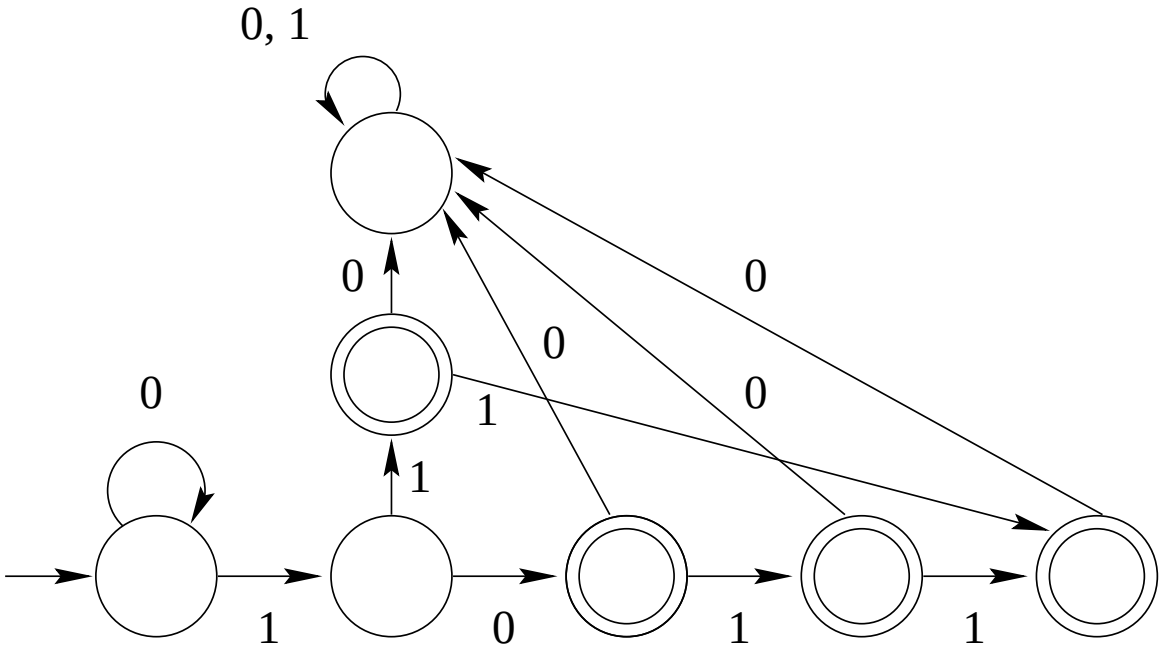
$$\begin{aligned} h(h^\omega(a)) &= h(a\ x\ h(x)\ h^2(x)\ h^3(x) \cdots) \\ &= h(a)\ h(x)\ h^2(x)\ h^3(x) \cdots \\ &= a\ x\ h(x)\ h^2(x)\ h^3(x) \cdots \\ &= h^\omega(a). \end{aligned}$$

Finite Automata

It turns out that in certain cases, the fixed points of morphisms can be described in an easy way.

The description involves finite automata.

- A *deterministic finite automaton* (DFA) is a simple model of a computer.



Transition diagram for automaton accepting the base-2 representations of the primes $p \leq 11$

Basics of Finite Automata

- Formally a DFA is a quintuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

  where:

  - Q is a finite set of *states*;
  - Σ is a finite set of symbols, the *input alphabet*;
  - $q_0 \in Q$ is the *start state*;
  - $F \subseteq Q$ is the set of *final states*;
  - $\delta : Q \times \Sigma \to Q$ is the *transition function*, which, given a state and an input symbol, tells which state the automaton should be in after reading the symbol.
  - the domain of δ can be extended to $\delta : Q \times \Sigma^*$ by (informally) defining $\delta(q, x)$ to be the state M is in after reading the string of symbols x, starting in state q.

Basics of Finite Automata

- The *language accepted by* M is denoted by $L(M)$ and is given by

$$\{w \in \Sigma^* \mid \delta(q_0, w) \in F\}.$$

- A language L is said to be *regular* if it is accepted by some DFA M.

Automata as Computers of Sequences

- We can generalize our notion of automaton to provide an output, not simply accept/reject.
- Formally, we define a *deterministic finite automaton with output* (DFAO) as a sextuple: $(Q, \Sigma, \delta, q_0, \Delta, \tau)$, where
  - Δ is the finite *output alphabet*; and
  - $\tau : Q \to \Delta$ is the *output mapping*.
- Next, we decide on a integer base $k \geq 2$ and represent n as a string of symbols over the alphabet $\Sigma = \{0, 1, 2, \ldots, k-1\}$.
- To compute f_n, given an automaton M, express n in base-k, say,

  $$a_r a_{r-1} \cdots a_1 a_0,$$

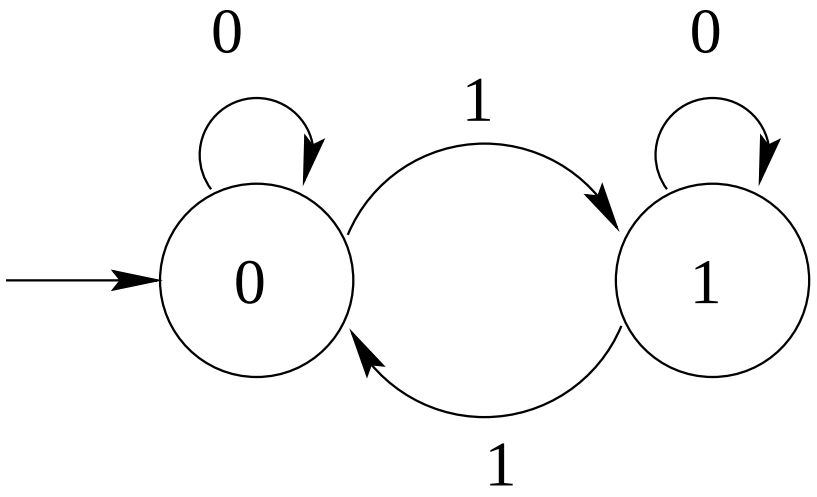
  and compute

  $$f_n = \tau(\delta(q_0, a_r a_{r-1} \cdots a_1 a_0)).$$

- Any sequence that can be computed in this way is said to be k-automatic.

Example: The Thue-Morse sequence

- The THUE-MORSE sequence $(t_n)_{n\geq 0}$ is defined as follows: t_n is the parity of the number of 1's in the binary expansion of n.
- $(t_n)_{n\geq 0} = 01101001100101 10\cdots$
- It is generated by the following DFAO:



For example, to compute t_{27}, we expand 27 in base 2 to get 11011, and then feed these digits into the automaton starting with the most significant digit.

We end up in a state labeled 0, so $t_{27} = 0$.

Example: The Rudin-Shapiro sequence

Here's another example of a 2-automatic sequence: the Rudin-Shapiro sequence.

The *Rudin-Shapiro* sequence $(r_n)_{n\geq 0}$, is defined by the formula $r_n = (-1)^{e_{2;11}(n)}$. In other words, r_n is 1 (resp. -1) according to whether the number of (possibly overlapping) occurrences of "11" in the binary expansion of n is even (resp. odd). Then $(r_n)_{n\geq 0}$ is 2-automatic, since it is generated by the DFAO in Figure 2.

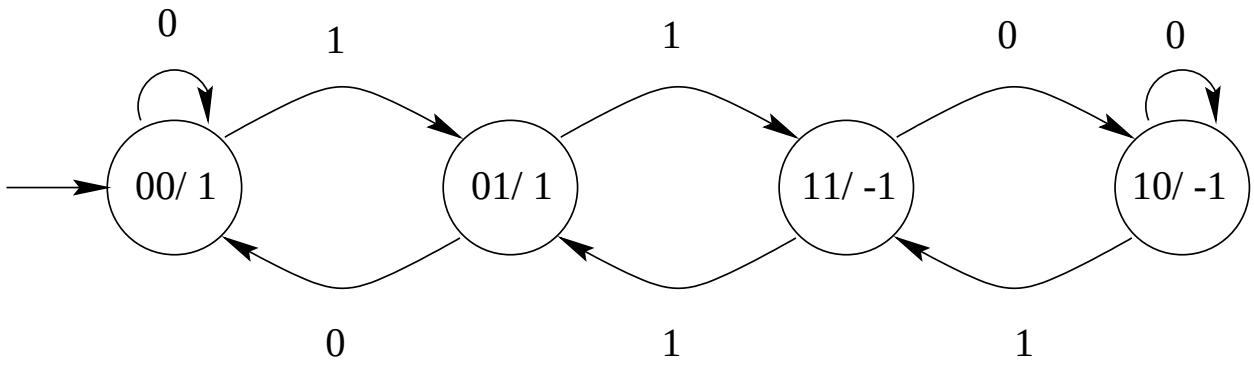


Figure 2: DFAO generating the Rudin-Shapiro sequence

Here the meaning of a state labeled ab/c is that the running sum of the number of occurrences of "11" so far is congruent to a modulo 2, the last digit input was b, and the output is c.

Here is another interesting feature of the Rudin-Shapiro sequence. Suppose we consider a path visiting lattice points in the plane. We start at the origin and make a first move to $(0,1)$. At step n, for $n \geq 1$, we decide to turn left (L) or right (R) according to the following rule:

$$d_n = \begin{cases} \mathtt{L}, & \text{if } r_n r_{n-1} = (-1)^n; \\ \mathtt{R}, & \text{if } r_n r_{n-1} = -(-1)^n. \end{cases} \qquad (1)$$

The first few terms of this sequence are given below:

n	$=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	..
r_n	$=$	1	1	1	-1	1	1	-1	1	1	1	1	-1	-1	-1	-1	-1	
d_n	$=$		R	L	L	R	R	R	L	L	R	L	L	L	R	R	L	R

We then get a space-filling curve that visits every lattice point in the $\frac{1}{8}$ of the plane; see the picture on the next slide.

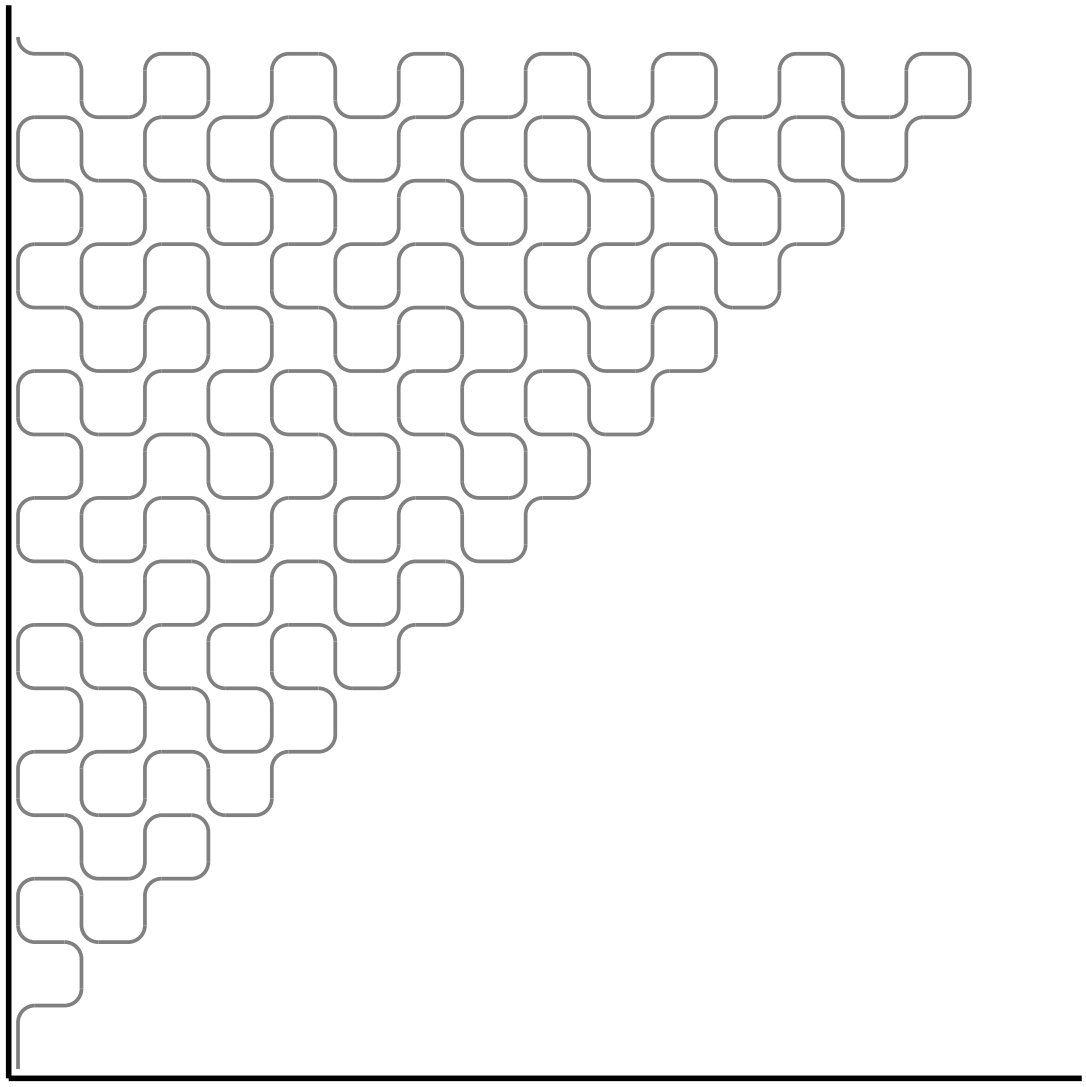


Figure 3: Space-filling curve derived from the Rudin-Shapiro sequence

Paperfolding

Back on Tuesday I told you about the paperfolding sequence.

We obtain this sequence by taking a rectangular piece of paper and folding it in half lengthwise, then folding the result in half again, etc., ad infinitum, taking care to make the folds in the same direction each time.

Next, we unfold the paper. The sequence $(f_i)_{i\geq 1}$ of "hills" $(+1)$ and "valleys" (-1) that results is called the *regular paperfolding sequence*.

For example, after one fold, and unfolding to $90°$, we get the pattern in Figure 4.

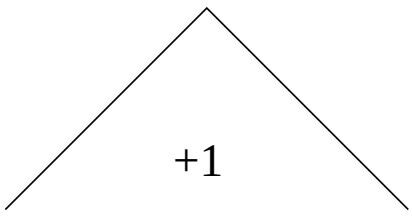


Figure 4: Building the regular paperfolding sequence: one fold

Paperfolding

After two folds, and unfolding to $90°$, we get the pattern in Figure 5.

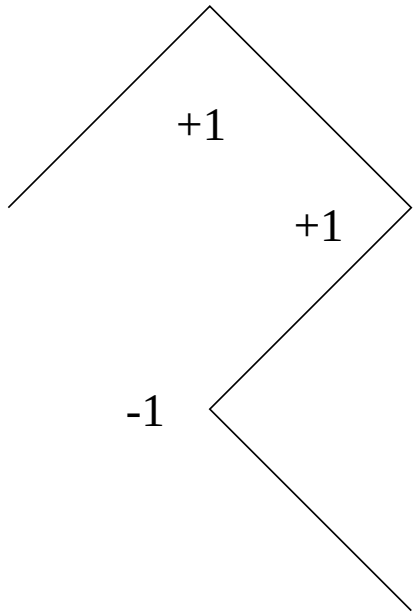


Figure 5: Building the regular paperfolding sequence: two folds

Here are the first few terms of the sequence $\mathbf{f}$:

$$n = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 1$$
$$f_n = 1\ 1\ -1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ 1\ -1\ -1\ 1$$

Paperfolding

For the regular paperfolding sequence we had the following formula:

$$\text{If } n = 2^k(2a+1) \text{ then } f_n = (-1)^a.$$

It follows that the regular paperfolding sequence $\mathbf{f} = (f_i)_{i \geq 1}$ is generated by the 2-DFAO in Figure 6.

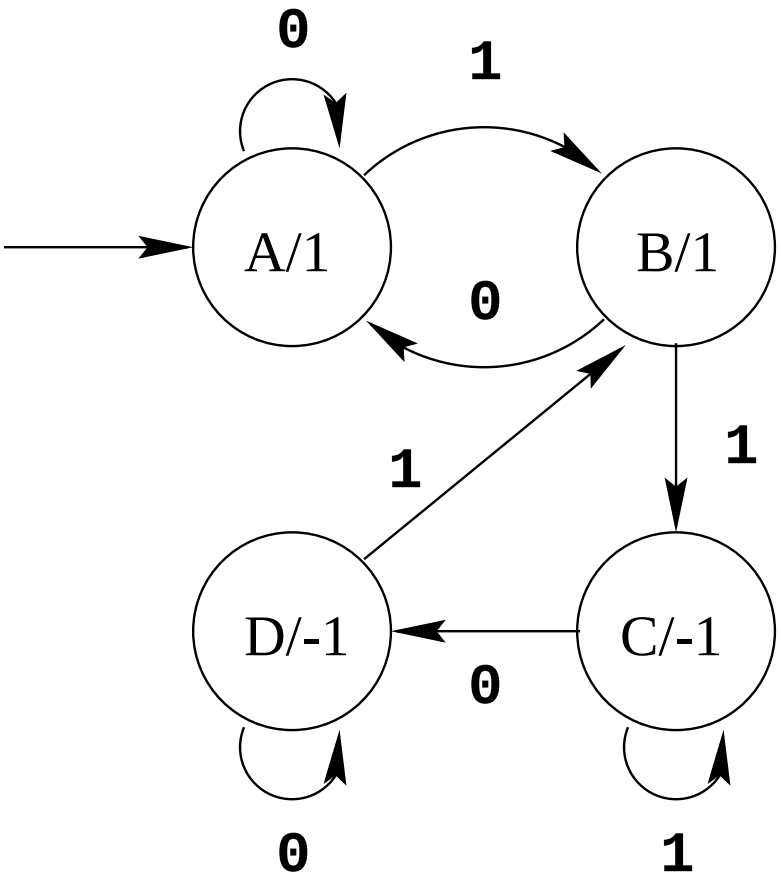


Figure 6: 2-DFAO for the paperfolding sequence

Cobham's Theorem

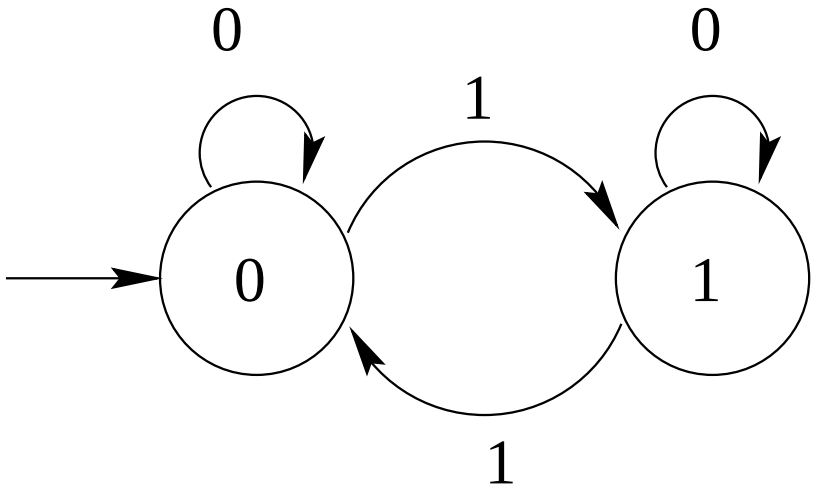
- A 1972 theorem of Alan Cobham says that a sequence is the pointwise image of a fixed point of a k-uniform morphism if and only if it is computed by a k-DFAO.
- By k-uniform morphism h, we mean h maps each letter to a string of k letters.
- By "pointwise image" we mean that we are allowed to rename the individual letters
- By a k-DFAO, we mean a finite automaton that accepts a number written in base-k as input, and outputs the name of the last state reached.

Examples of Cobham's Theorem

The Thue-Morse sequence is generated as the fixed point of the Thue-Morse morphism

$$\mu(0) = 01$$
$$\mu(1) = 10.$$

It is also generated by the following 2-DFAO:



Examples of Cobham's Theorem

The Rudin-Shapiro sequence is generated as the image under the map

$$\tau(\mathtt{A}) = 1 \qquad \tau(\mathtt{C}) = -1$$
$$\tau(\mathtt{B}) = 1 \qquad \tau(\mathtt{D}) = -1$$

of the fixed point, starting with A, of the morphism

$$h(\mathtt{A}) = \mathtt{AB}$$
$$h(\mathtt{B}) = \mathtt{AC}$$
$$h(\mathtt{C}) = \mathtt{DB}$$
$$h(\mathtt{D}) = \mathtt{DC}$$

It is also generated by the following 2-DFAO:

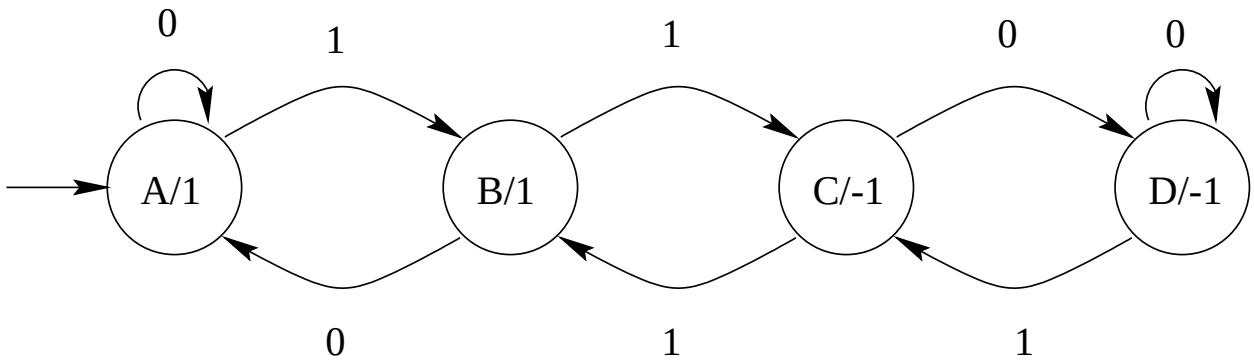


Figure 7: DFAO generating the Rudin-Shapiro sequence

Examples of Cobham's Theorem

The regular paperfolding sequence is generated as the image under the map

$$\tau(\mathtt{A}) = 1 \qquad \tau(\mathtt{C}) = -1$$
$$\tau(\mathtt{B}) = 1 \qquad \tau(\mathtt{D}) = -1$$

of the fixed point, starting with A, of the morphism

$$h(\mathtt{A}) = \mathtt{AB} \qquad h(\mathtt{C}) = \mathtt{DC}$$
$$h(\mathtt{B}) = \mathtt{AC} \qquad h(\mathtt{D}) = \mathtt{DB}$$

It is also generated by the following 2-DFAO:

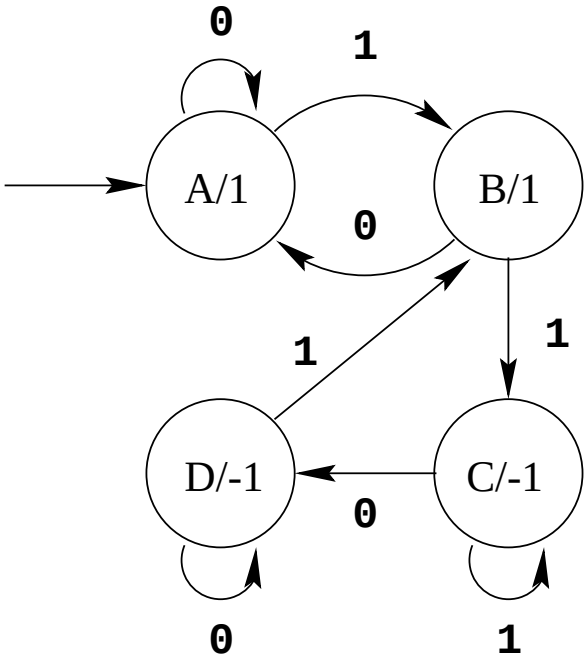


Figure 8: 2-DFAO for the paperfolding sequence

The Tower of Hanoi Puzzle

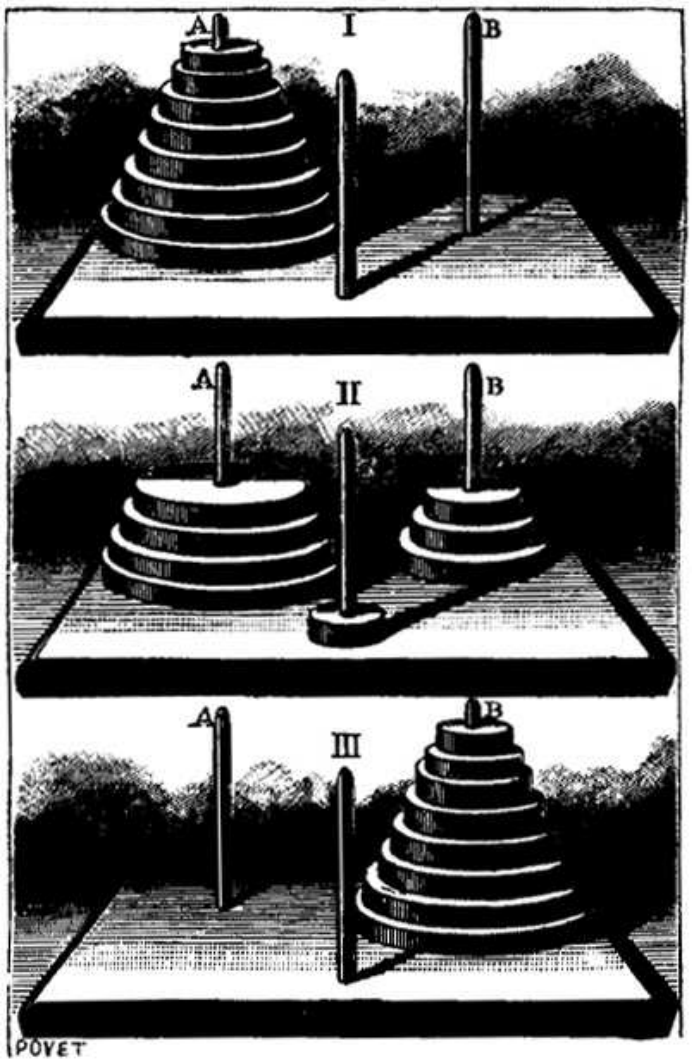


Figure 9: The Tower of Hanoi Puzzle

- The Tower of Hanoi puzzle consists of
  - three numbered pegs
  - N disks

Tower of Hanoi Puzzle: Rules

- Initially, the disks, which have radius $1, 2, \ldots, N$, are all placed on page 1 in increasing order of size (smallest disk at top; largest disk at bottom)
- A *move* of the puzzle consists of taking the top disk off one peg and moving it to another
- A move is called *legal* if it does not involve covering a smaller disk with a larger one.
- The object of the puzzle is to transfer all of the disks from peg 1 to one of the other two pegs, using only legal moves.
- We are most interested in *optimal* solutions to the puzzle; that is, solutions for N disks that use the *smallest* number of legal moves.

The Tower of Hanoi Puzzle: Coding the Moves

- At each step there are six possible moves.
- We can code the moves as follows:
  - $\mathtt{a}$ means 'move disk from peg 1 to peg 2';
  - $\mathtt{b}$ means 'move disk from peg 2 to peg 3';
  - $\mathtt{c}$ means 'move disk from peg 3 to peg 1';
  - $\overline{\mathtt{a}}$ means 'move disk from peg 2 to peg 1';
  - $\overline{\mathtt{b}}$ means 'move disk from peg 3 to peg 2';
  - $\overline{\mathtt{c}}$ means 'move disk from peg 1 to peg 3'.
- Note: $\overline{\mathtt{a}}$ is the inverse of $\mathtt{a}$, etc.
- Thus, for example, we can move three disks from peg 1 to peg 2 as follows: $\mathtt{a}\ \overline{\mathtt{c}}\ \mathtt{b}\ \mathtt{a}\ \mathtt{c}\ \overline{\mathtt{b}}\ \mathtt{a}$.
- This is an optimal solution for three disks.

The Tower of Hanoi: Optimal Solutions

- The Tower of Hanoi puzzle with N disks can be solved in $2^N - 1$ steps, by a recursive procedure:
  - First, move $N-1$ disks from peg 1 to peg 3 (recursively);
  - Now move disk N to peg 2;
  - Finally, move the $N-1$ disks on peg 3 to peg 2 (recursively).
- Let T_N be the number of steps in this solution.
- Then $T_N = 2T_{N-1} + 1$. Since $T_0 = 0$, it follows that $T_N = 2^N - 1$.
- But is this solution best possible?

The Tower of Hanoi: Optimal Solutions

- The recursive solution we just saw is actually best possible, because
  - In order to transfer N disks from one peg to another, we must at some point move disk N at least once;
  - In order to move disk N (which is the largest), it must be alone on its peg, and some other peg must be empty; hence the remaining peg must contain the $N-1$ smaller disks;
  - Hence no optimal strategy can do better than to

    (i) move the $N-1$ smaller disks;

    (ii) move disk N;

    (iii) then move the $N-1$ smaller disks again.
- It follows that if R_N denotes the number of moves used by the optimal strategy for N disks, then $R_N \geq 2R_{N-1}+1$. Hence $R_N \geq 2^N - 1$.

The Tower of Hanoi Inventor: Edouard Lucas

Figure 10: Edouard Lucas (1842–1891)

- Edouard Lucas (1842–1891) was a French number theorist and recreational mathematician
- He invented the Tower of Hanoi puzzle c. 1883, and attributed it to "N. Claus (de Siam)" (an anagram for Lucas) of the college of "Li-Sou-Stian" (anagram for Saint-Louis).

The Tower of Hanoi Puzzle: Coding the Optimal Solutions

- We consider the optimal solutions to the N-disk puzzle that move
  - N disks from peg 1 to peg 2 if N is odd;
  - N disks from peg 1 to peg 3 if N is even.
- Unnatural? But has the advantage that the sequence representing the optimal solution for $N+1$ disks begins with the sequence for N disks.
- Now code the optimal solution for N disks by a string of symbols H_N. For example:
  - $H_0 = \epsilon$ (the empty string)
  - $H_1 = \mathtt{a}$
  - $H_2 = \mathtt{a}\ \overline{\mathtt{c}}\ \mathtt{b}$
  - $H_3 = \mathtt{a}\ \overline{\mathtt{c}}\ \mathtt{b}\ \mathtt{a}\ \mathtt{c}\ \overline{\mathtt{b}}\ \mathtt{a}$
- Hence there is actually an *infinite* string of symbols

$$H = h_0 h_1 h_2 \cdots = \mathtt{a}\,\overline{\mathtt{c}}\,\mathtt{b}\,\mathtt{a}\,\mathtt{c}\,\overline{\mathtt{b}}\,\mathtt{a}\,\overline{\mathtt{c}}\,\mathtt{b}\,\overline{\mathtt{a}}\,\mathtt{c}\,\mathtt{b}\,\mathtt{a}\,\overline{\mathtt{c}}\,\mathtt{b}\,\cdots$$

which codes the solution for $N = 1, 2, 3, \ldots$ disks.

- Another way to think of H is that it solves the puzzle for an infinite number of disks!

The Tower of Hanoi Puzzle: Coding the Optimal Solutions

- The infinite string H can be described as the limit of the sequence of strings $(H_i)_{i \geq 0}$, where each H_i is a string of length $2^i - 1$ giving the solution to the puzzle for i disks.
- Then we have:

$$\begin{aligned} H_0 &= \epsilon; \\ H_{2N} &= H_{2N-1}\ \overline{\mathsf{c}}\ \sigma(H_{2N-1}), \quad \text{for } N \geq 1; \\ H_{2N+1} &= H_{2N}\ \mathsf{a}\ \sigma^{-1}(H_{2N}), \quad \text{for } N \geq 0. \end{aligned}$$

- Recall: H_{2N} moves disks from peg 1 to peg 3
- H_{2N+1} moves disks from peg 1 to peg 2
- Here σ is a morphism defined as follows:

$$\begin{aligned} \sigma(\mathsf{a}) &= \mathsf{b}; & \sigma(\overline{\mathsf{a}}) &= \overline{\mathsf{b}}\ ; \\ \sigma(\mathsf{b}) &= \mathsf{c}; & \sigma(\overline{\mathsf{b}}) &= \overline{\mathsf{c}}\ ; \\ \sigma(\mathsf{c}) &= \mathsf{a}; & \sigma(\overline{\mathsf{c}}) &= \overline{\mathsf{a}}\ . \end{aligned}$$

- Note that $\lim_{N \to \infty} H_N = H$.

A Useful Lemma

Lemma 1. *We have*

$$H = (\mathtt{a\ C\ b\ A\ c\ B})^{\infty},$$

where

A *means* a or $\overline{\mathtt{a}}$;
B *means* b or $\overline{\mathtt{b}}$;
C *means* c or $\overline{\mathtt{c}}$.

Proof. Prove, by induction on N, that there exists e such that

$$H_{2N+1} = (\mathtt{a\ C\ b\ A\ c\ B})^{e}\ \mathtt{a}\ ;$$
$$H_{2N} = (\mathtt{a\ C\ b\ A\ c\ B})^{e}\ \mathtt{a\ C\ b}\ .$$

For example,

$$H_{2N+1} = H_{2N}\ \mathtt{a}\ \sigma^{-1}(H_{2N})$$

$$= (\mathtt{a\ C\ b\ A\ c\ B})^{e}\ \mathtt{a\ C\ b\ a}\ (\mathtt{c\ B\ a\ C\ b\ A})^{e}\ \mathtt{c\ B\ a}$$

$$= (\mathtt{a\ C\ b\ A\ c\ B})^{2e+1}\ \mathtt{a}.$$

The Tower of Hanoi Sequence H Via Iterated Morphisms

- Define

$$\begin{aligned} \varphi(\mathtt{a}) &= \mathtt{a}\ \overline{\mathtt{c}}; & \varphi(\overline{\mathtt{a}}) &= \mathtt{a}\ \mathtt{c}\ ; \\ \varphi(\mathtt{b}) &= \mathtt{c}\ \overline{\mathtt{b}}; & \varphi(\overline{\mathtt{b}}) &= \mathtt{c}\ \mathtt{b}\ ; \\ \varphi(\mathtt{c}) &= \mathtt{b}\ \overline{\mathtt{a}}; & \varphi(\overline{\mathtt{c}}) &= \mathtt{b}\ \mathtt{a}\ . \end{aligned}$$

- Then $H = \varphi(H)$.
- For example:

$$\begin{aligned} H &= \mathtt{a}\ \ \overline{\mathtt{c}}\ \ \mathtt{b}\ \ \mathtt{a}\ \ \mathtt{c}\ \ \overline{\mathtt{b}}\ \ \mathtt{a}\ \cdots \\ \varphi(H) &= \mathtt{a}\overline{\mathtt{c}}\ \mathtt{ba}\ \mathtt{c}\overline{\mathtt{b}}\ \mathtt{a}\overline{\mathtt{c}}\ \mathtt{b}\overline{\mathtt{a}}\ \mathtt{cb}\ \mathtt{a}\overline{\mathtt{c}}\ \cdots \end{aligned}$$

Another Useful Lemma

Recall

$$\begin{aligned} \varphi(\mathsf{a}) &= \mathsf{a}\ \overline{\mathsf{c}}; & \varphi(\overline{\mathsf{a}}) &= \mathsf{a}\ \mathsf{c}\ ; \\ \varphi(\mathsf{b}) &= \mathsf{c}\ \overline{\mathsf{b}}; & \varphi(\overline{\mathsf{b}}) &= \mathsf{c}\ \mathsf{b}\ ; \\ \varphi(\mathsf{c}) &= \mathsf{b}\ \overline{\mathsf{a}}; & \varphi(\overline{\mathsf{c}}) &= \mathsf{b}\ \mathsf{a}\ . \end{aligned}$$

Also

$$\begin{aligned} \sigma(\mathsf{a}) &= \mathsf{b}; & \sigma(\overline{\mathsf{a}}) &= \overline{\mathsf{b}}\ ; \\ \sigma(\mathsf{b}) &= \mathsf{c}; & \sigma(\overline{\mathsf{b}}) &= \overline{\mathsf{c}}\ ; \\ \sigma(\mathsf{c}) &= \mathsf{a}; & \sigma(\overline{\mathsf{c}}) &= \overline{\mathsf{a}}\ . \end{aligned}$$

Lemma 2. We have

$$\begin{aligned} \varphi(\sigma(w)) &= \sigma^{-1}(\varphi(w)); \\ \varphi(\sigma^{-1}(w)) &= \sigma(\varphi(w)). \end{aligned}$$

Proof. Verify both equations on each of the six

letters. For example:

$$\begin{aligned}\varphi(\sigma(\mathtt{a})) &= \varphi(\mathtt{b}) \\ &= \mathtt{c}\ \overline{\mathtt{b}} \\ &= \sigma^{-1}(\mathtt{a}\ \overline{\mathtt{c}}) \\ &= \sigma^{-1}(\varphi(\mathtt{a})).\end{aligned}$$

H is a Fixed Point of φ

Theorem. *We have* $H = \varphi(H)$.

Proof. It suffices to show

$$\begin{aligned} \varphi(H_{2N})\mathtt{a} &= H_{2N+1}; \\ \varphi(H_{2N+1})\mathtt{b} &= H_{2N+2}. \end{aligned}$$

This can be shown by induction on N. For example:

$$\begin{aligned} \varphi(H_{2N})\mathtt{a} &= \varphi(H_{2N-1}\ \overline{\mathtt{c}}\ \sigma(H_{2N-1}))\ \mathtt{a} \\ &= \varphi(H_{2N-1})\ \mathtt{b}\ \mathtt{a}\ \varphi(\sigma(H_{2N-1}))\ \mathtt{a} \\ &= \varphi(H_{2N-1})\ \mathtt{b}\ \mathtt{a}\ \sigma^{-1}(\varphi(H_{2N-1}))\ \mathtt{a} \\ &= \varphi(H_{2N-1}\ \mathtt{b})\ \mathtt{a}\ \sigma^{-1}(\varphi(H_{2N-1})\ \mathtt{b}) \\ &= H_{2N}\ \mathtt{a}\ \sigma^{-1}(H_{2N}) \\ &= H_{2N+1}. \end{aligned}$$

It follows that H is a fixed point of φ.

Another Way to Generate H

We can also generate H by iterating the morphism φ:

$$\begin{aligned}
\varphi(\mathsf{a}) &= \mathsf{a}\ \overline{\mathsf{c}} = H_1\ \overline{\mathsf{c}} \\
\varphi^2(\mathsf{a}) &= \mathsf{a}\ \overline{\mathsf{c}}\ \mathsf{b}\ \mathsf{a} = H_2\ \mathsf{a} \\
\varphi^3(\mathsf{a}) &= \mathsf{a}\ \overline{\mathsf{c}}\ \mathsf{b}\ \mathsf{a}\ \mathsf{c}\ \overline{\mathsf{b}}\ \mathsf{a}\ \overline{\mathsf{c}} = H_3\ \overline{\mathsf{c}} \\
&\vdots
\end{aligned}$$

In fact, one can show by induction that

$$\begin{aligned}
\varphi^{2N}(\mathsf{a}) &= H_{2N}\ \mathsf{a}; \\
\varphi^{2N+1}(\mathsf{a}) &= H_{2N+1}\ \overline{\mathsf{c}}.
\end{aligned}$$

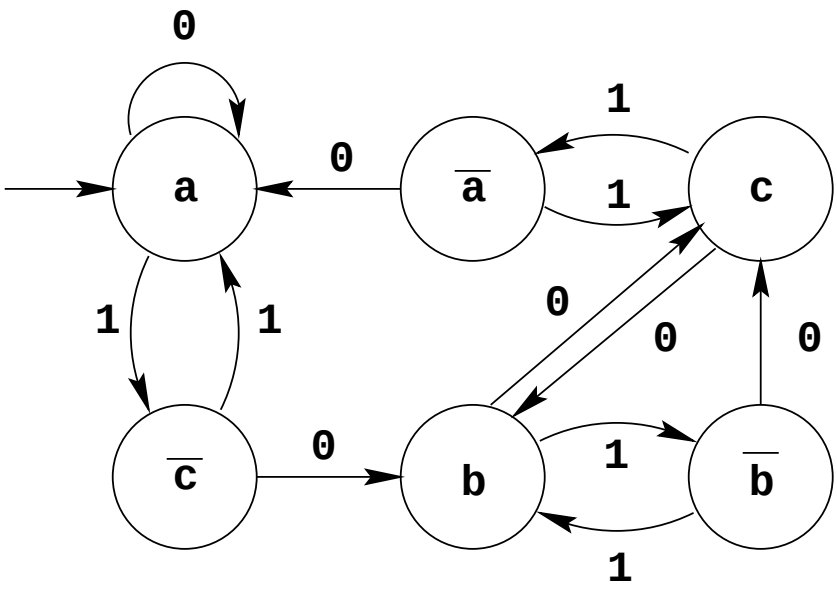


Figure 11: The Tower of Hanoi automaton

- We can use this automaton to compute the billionth term of the sequence H.
- We have

$$1,000,000,000_{(2)} =$$

$$11101\ 11001\ 10101\ 10010\ 10000\ 00000.$$

- The billionth term is c.

Thue's Problem

- A string y is a *subword* of a string u if we can write $u = xyz$ for some strings x, z.
- A string u is *squarefree* if it contains no subword of the form yy, where y is nonempty.
- A string u is *cubefree* if it contains no subword of the form yyy, where y is nonempty.
- Axel Thue (c. 1906) constructed an *infinite* squarefree string over $\{0, 1, 2\}$ and an infinite cubefree string over $\{0, 1\}$.

Figure 12: Axel Thue (1863–1922)

Squarefree Strings and the Tower of Hanoi

Conjecture. (Fred Dixon) *The Tower of Hanoi sequence H is squarefree.*

Proof. (Dan Astoorian & Jim Randall)

We need some lemmas.

Lemma 3. *Ignoring the bars, the symbols of H are periodic with period 3.*

Proof. Follows from Lemma 1.

Lemma 4. *If $H = h_0 h_1 h_2 \cdots$, then*

$$h_i = \begin{cases} \mathsf{a}, & \text{if } i \equiv 0 \pmod 6; \\ \mathsf{b}, & \text{if } i \equiv 2 \pmod 6; \\ \mathsf{c}, & \text{if } i \equiv 4 \pmod 6. \end{cases}$$

Hence the symbols occurring in even-numbered positions of H are "unbarred".

Proof. Follows from Lemma 1.

Squarefree Strings and the Tower of Hanoi

Lemma 5. *H doesn't contain 4 consecutive unbarred symbols.*

Proof. If it did, say $h_i h_{i+1} h_{i+2} h_{i+3}$, then i is either even or odd.

If i is even, say $i = 2k$, then

$$\varphi(h_k h_{k+1}) = h_i h_{i+1} h_{i+2} h_{i+3}.$$

Then both h_k and h_{k+1} have to be barred, contradicting Lemma 4.

If i is odd, say $i = 2k + 1$, then h_{i-1} must be unbarred, so we can repeat the same argument on $h_{i-1} h_i h_{i+1} h_{i+2}$.

Proof that H is Squarefree

Suppose H did contain a square subword

$$xx = x_1x_2x_3\cdots x_n\ x_1x_2x_3\cdots x_n.$$

Assume that

(i) xx is as short as possible; and

(ii) among all squares with this length in H, xx occurs earliest in H.

Case A. $n = |x|$ is odd. Then, by Lemma 4, all the symbols of x are unbarred. If $n \geq 2$, this gives 4 consecutive unbarred symbols, a contradiction. If $n = 1$, this gives $xx \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, which cannot happen.

Case B. $n = |x| = 2m$, and xx begins at an even-numbered position in H, say $2j$. Then

$$xx = \underbrace{h_{2j}h_{2j+1}\cdots h_{2j+2m-1}}_{x}\cdot \underbrace{h_{2j+2m}h_{2j+2m+1}\cdots h_{2j+4m-1}}_{x}$$

and

$$\varphi(h_j h_{j+1}\cdots h_{j+2m-1}) = xx.$$

Hence

$$x = \varphi(h_j \cdots h_{j+m-1})$$

and

$$x = \varphi(h_{j+m} \cdots h_{j+2m-1}),$$

and so

$$h_j \cdots h_{j+m-1} = h_{j+m} \cdots h_{j+2m-1}.$$

Thus $h_j \cdots h_{j+2m-1} = yy$, and $|y| < |x|$, a contradiction.

Case C. $n = |x|$ is even, and xx begins at an odd-numbered position in H, say $2j + 1$.

Then

$$qxx = h_{2j}\ h_{2j+1} \cdots h_{2j+n}\ h_{2j+n+1} \cdots h_{2j+2n}$$

for some symbol $q = h_{2j}$.

By Lemma 3, we have $3 \mid n$. Hence $6 \mid n$. Hence $q = h_{2j+n}$.

Thus

$$\underbrace{h_{2j}h_{2j+1} \cdots h_{2j+n-1}}\underbrace{h_{2j+n}h_{2j+n+1} \cdots h_{2j+2n-1}}$$

is a square of the same length as xx, but occurring one position earlier in H, a contradiction.

The proof is complete.

An Open Problem

Believe it or not, the optimal solution for the Tower of Hanoi with **4** pegs is still not known.

Morphisms and Computer Graphics

Consider the morphism f defined as follows:

$$\begin{aligned}
\mathrm{S}_1 &\to \mathrm{S}_1\mathrm{R}_1\mathrm{R}_2 \\
\mathrm{S}_2 &\to \mathrm{S}_2\mathrm{L}_1\mathrm{L}_2 \\
\mathrm{R}_1 &\to \mathrm{S}_2\mathrm{L}_1\mathrm{L}_2 \\
\mathrm{R}_2 &\to \mathrm{S}_1\mathrm{S}_2\mathrm{S}_1 \\
\mathrm{L}_1 &\to \mathrm{S}_1\mathrm{R}_1\mathrm{R}_2 \\
\mathrm{L}_2 &\to \mathrm{S}_2\mathrm{S}_1\mathrm{S}_2
\end{aligned}$$

and let τ be a coding that removes the subscripts. Let $w_i = \tau(f^{2i}(\mathrm{S}_1))$ with the last two characters removed, and interpret the letters in w_i as drawing instructions, where S means move straight one unit in the direction you are already traveling, R means turn right and move one unit, and L means turn left and move one unit. Then under this interpretation, the words w_i code the i'th iteration in the construction of Peano's famous space-filling curve.

Morphisms and Computer Graphics

For example, Figure 13 shows the graphic interpretation of w_3:

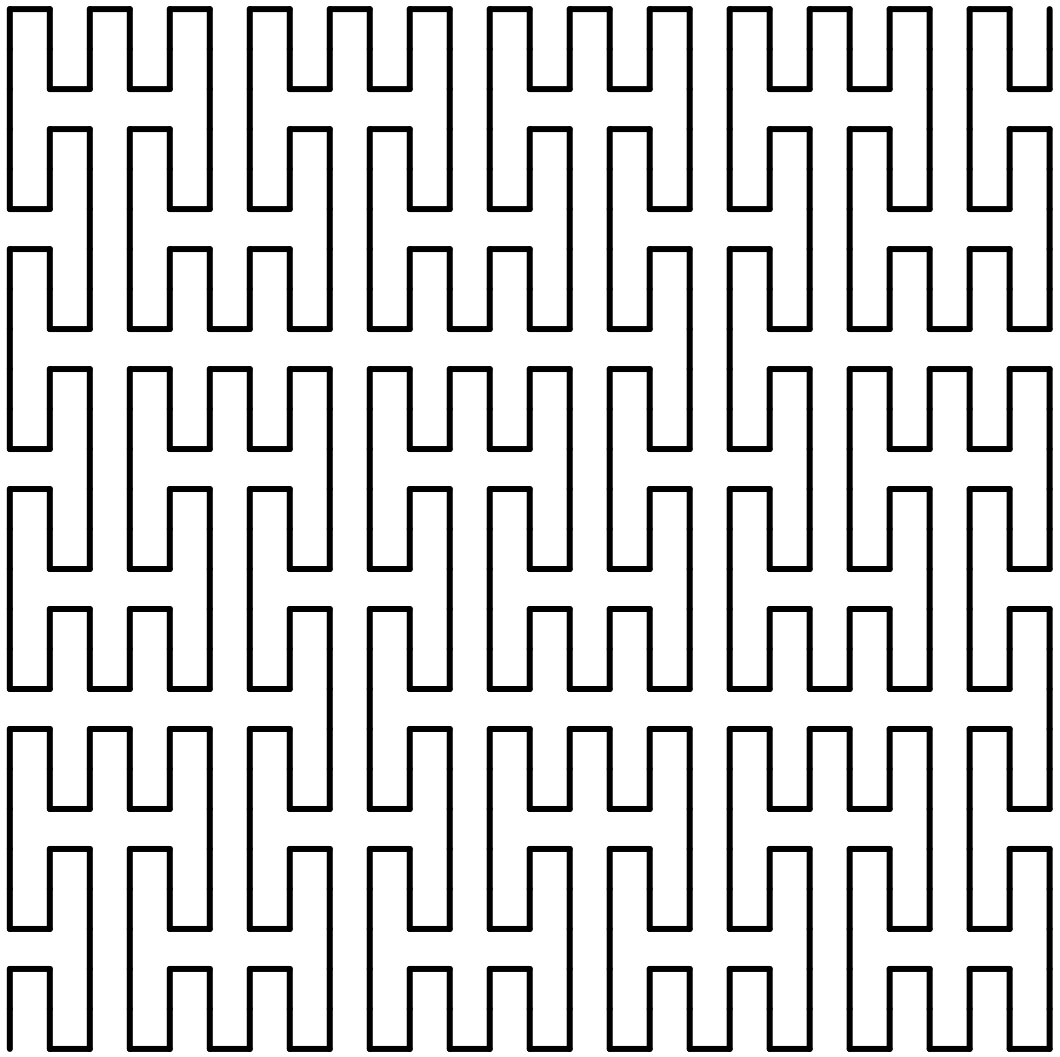


Figure 13: Third iteration in the construction of Peano's spacefilling curve

Morphisms and Computer Graphics

Consider the morphism g defined as follows:

$$\begin{aligned} \mathtt{R} &\to \mathtt{RLLSRRLR} \\ \mathtt{L} &\to \mathtt{RLLSRRLL} \\ \mathtt{S} &\to \mathtt{RLLSRRLS}. \end{aligned}$$

Define $x_i = g^i(\mathtt{RRRR})$, and consider the figures specified by x_i, where the interpretations are as in the previous slide.

Then x_i codes the i'th iteration in the construction of von Koch's famous snowflake curve. The first four iterations are shown in Figure 14.

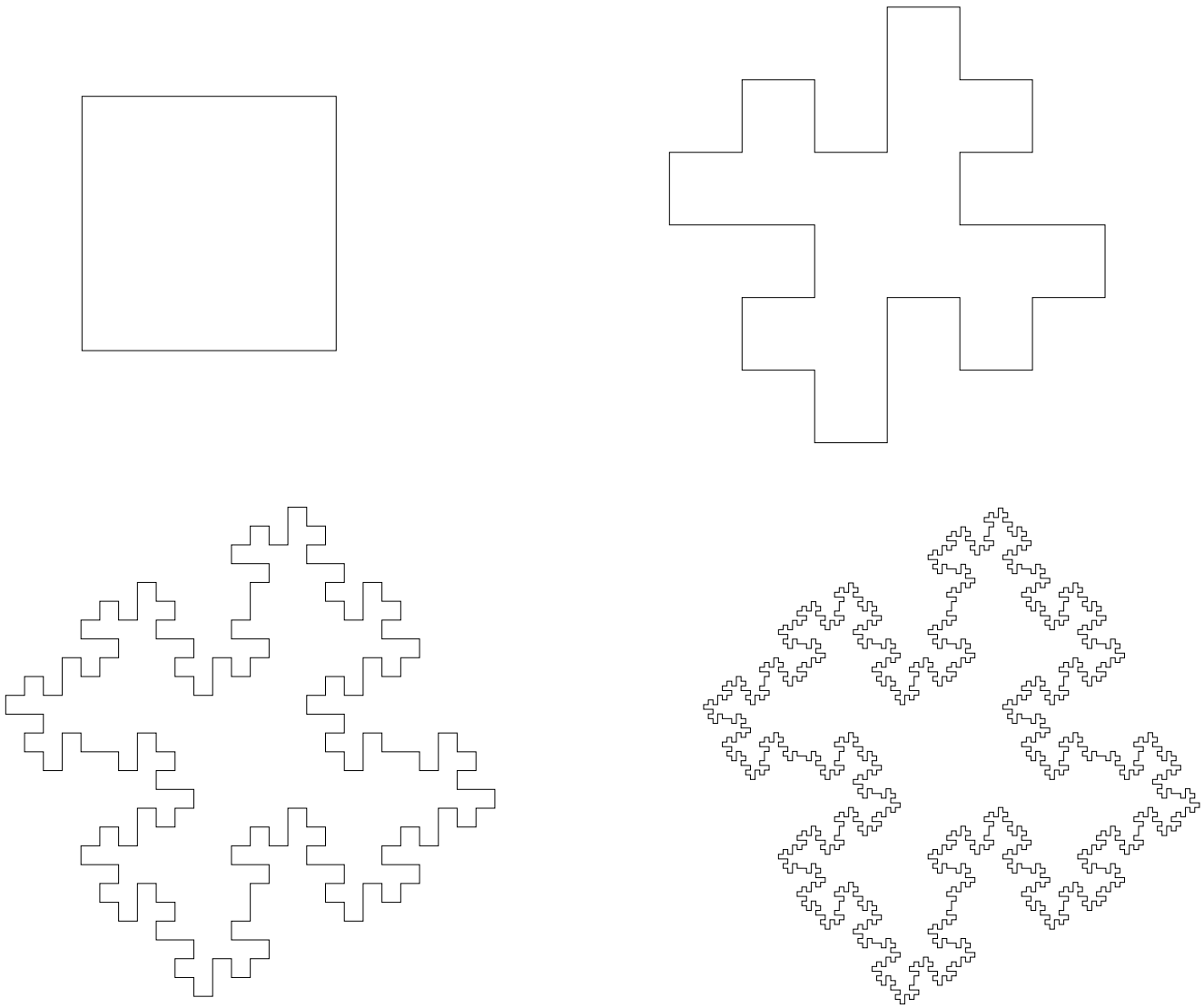


Figure 14: Four iterations in the construction of the von Koch curve

The Christol Theorem

As we have seen, k-automatic sequences can be generated in two different ways:

- As the pointwise image of a fixed point of a k-uniform morphism;
- As the sequence generated by a k-DFAO.

There is a third way to characterize these sequences, discovered by the French mathematician Gilles Christol in 1979.

Figure 15: Gilles Christol

The Christol Theorem

Theorem.

Let p be a prime number.

Let $(a_i)_{i \geq 0}$ be a sequence taking values in

$$\{0, 1, \ldots, p-1\}.$$

Then $(a_i)_{i \geq 0}$ is p-automatic if and only if the formal Laurent series

$$\sum_{i \geq 0} a_i X^{-i}$$

is algebraic over $GF(p)(X)$.

Example. Recall what we proved about the Thue-Morse series

$$F(X) = \sum_{i \geq 0} t_i X^{-i}.$$

We have

$$(1+X)^3 F^2 + X(1+X)^2 F + X^2 = 0. \quad (2)$$

Abelian Square-Free Strings

Recall that we saw how to construct an infinite square-free string

$$2102012101202102012 02\cdots$$

on the alphabet $\{0, 1, 2\}$. This string has the property that it contains no subword of the form xx.

We can also consider avoiding abelian squares. An *abelian square* is a word of the form xx' where x' is a permutation (rearrangement) of the letters of x. For example,

$$21011021$$

is an abelian square because 1021 is a permutation of 2101.

Abelian Square-Free Strings

It's not hard to show that every word of length 8 over a 3-letter alphabet contains an abelian square.

In 1961 Paul Erdős asked if there was an infinite abelian square-free word over a 4-letter alphabet.

Figure 16: Paul Erdős (1913–1996)

Abelian Square-Free Strings

This problem remained open until 1992, when Veikko Keränen showed the fixed point of the following 85-uniform morphism was abelian square-free:

$$
\begin{aligned}
\mathtt{a} \to\ & \mathtt{abcacdcbcdcadcdbdabacabadbabc} \\
& \mathtt{bdbcbacbcdcacbabdabacadcbcdca} \\
& \mathtt{cdbcbacbcdcacdcbdcdadbdcbca} \\
\mathtt{b} \to\ & \mathtt{bcdbdadcdadbadacabcbdbcbacbcd} \\
& \mathtt{cacdcbdcdadbdcbcabcbdbadcdadb} \\
& \mathtt{dacdcbdcdadbdadcadabacadcdb} \\
\mathtt{c} \to\ & \mathtt{cdacabadabacbabdbcdcacdcbdcda} \\
& \mathtt{dbdadcadabacadcdbcdcacbadabac} \\
& \mathtt{abdadcadabacabadbabcbdbadac} \\
\mathtt{d} \to\ & \mathtt{dabdbcbabcbdcbcacdadbdadcadab} \\
& \mathtt{acabadbabcbdbadacdadbdcbabcbd} \\
& \mathtt{bcabadbabcbdbcbacbcdcacbabd}
\end{aligned}
$$

Tomi Laakso has set this to music.

Fixed Points of Transducers

Instead of iterating morphisms, we can consider a generalization: iterating outputs of finite-state transducers.

A finite-state transducer is like a finite automaton, but it outputs a string for each transition (instead of outputs being associated with states).

Consider the following transducer:

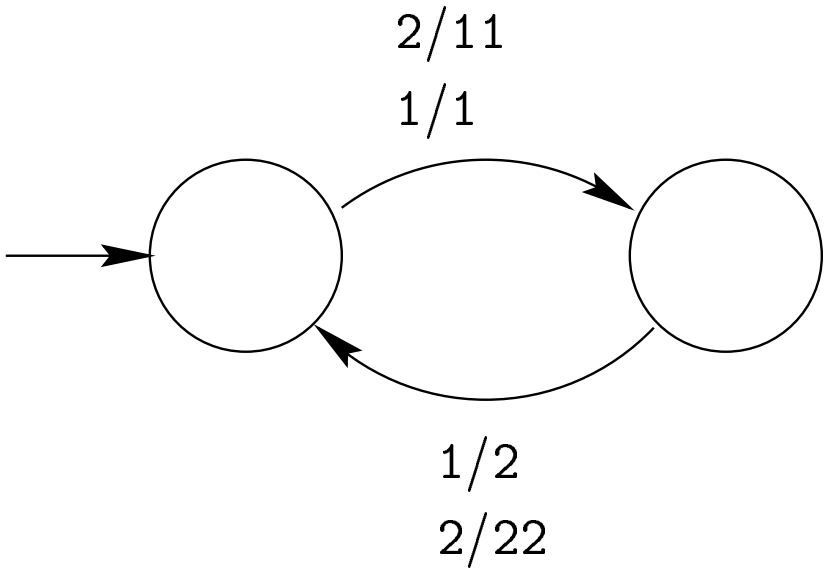


Figure 17: Transducer for the Kolakoski sequence

This transducer defines a map $k : \{1,2\}^* \to \{1,2\}^*$. For example, $k(221) = 11221$.

We can now iterate this map, starting with 2. We find:

$$
\begin{aligned}
k^0(2) &= 2 \\
k^1(2) &= 11 \\
k^2(2) &= 12 \\
k^3(2) &= 122 \\
k^4(2) &= 12211 \\
k^5(2) &= 1221121 \\
k^6(2) &= 1221121221 \\
k^7(2) &= 122112122122112 \\
&\vdots
\end{aligned}
$$

There is a unique infinite word

$$\mathbf{K} = 122112122122112\cdots$$

of which $k^0(2), k^1(2), \ldots$ are all prefixes. This is called the *Kolakoski sequence*.

Here is one beautiful property of the Kolakoski sequence: the sequence of run lengths of the sequence (lengths of blocks of consecutive identical digits) is the sequence itself.

Despite much work, very little is known about this sequence. For example, do the symbols 1 and 2 occur with a well-defined limiting frequency? And is this frequency $1/2$?

For Further Reading

1. A. Cobham, Uniform tag sequences, *Math. Systems Theory* **6** (1972), 164–192.

2. J.-P. Allouche and F. Dress, Tour de Hanoi et automates, *RAIRO Informatique Théorique et Applications*, **24** (1990), 1–15.

3. J.-P. Allouche, D. Astoorian, J. Randall, and J. Shallit, Morphisms, squarefree strings, and the Tower of Hanoi puzzle, *Amer. Math. Monthly* **101** (1994), 651–658.

4. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1990.