The tower of Hanoi

The tower of Hanoi consists of 3 pegs, labeled 1, 2, 3 and N disks of unequal sizes. Disks can be moved from peg to peg, but one can never put a larger disk on top of a smaller one. Initially, N disks are on peg 1 and we wish to move them to another peg.

Q1: What is the most efficient way to do this (using the smallest number of moves)?

It is easy to see the answer is $2^N - 1$. For in order to move all N disks, the Nth must move at some point. Letting $t(N)$ denote the number of moves to solve the puzzle in the most efficient way possible we see that to even be able to move the Nth disk, at least one peg must be empty, so all $N-1$ must be on the other. Once we move the Nth, we use the most efficient way to move the $N-1$ on top of it. This gives

$$t(N) = t(N-1) + 1 + t(N-1)$$

And since $t(0) = 0$ the result follows.
Now let's encode the moves as follows:

\[ a : 1 \to 2 \quad \bar{a} : 2 \to 1 \]
\[ b : 2 \to 3 \quad \bar{b} : 3 \to 2 \]
\[ c : 3 \to 1 \quad \bar{c} : 1 \to 3 \]

So to move 3 disks from peg 1 to peg 3 we write

\[ a \bar{c} b a c \bar{b} a \]

Now we introduce a coding that does a cyclic shift of the moves:

\[ \sigma(a) = b \quad \sigma(\bar{a}) = \bar{b} \]
\[ \sigma(b) = c \quad \sigma(\bar{b}) = \bar{c} \]
\[ \sigma(c) = a \quad \sigma(\bar{c}) = \bar{a} \]

Now define \( H_i \) to be the solution of the Hanoi problem that moves 3 disks from peg 1 to peg 2 if \( i \) is odd

\[ \text{peg 1 to peg 2 if } i \text{ is odd} \]

peg 1 to peg 3 if \( i \) is even

We define it this way so that \( H_i \) is a prefix of \( H_{i+1} \). Let \( H = \lim_{n \to \infty} H_n \), be unique infinite word at which \( H_1, H_2, H_3, \ldots \) are all prefixes.

\[ H_0 = \epsilon \]
\[ H_1 = a \]
\[ H_2 = a \bar{c} b \]

etc.
Lemma:

\[ H_{2i+1} = H_{2i} \circ \sigma^{-1} (H_{2i}) \quad i \geq 0 \quad (\checkmark) \]

\[ H_{2i} = H_{2i-1} \circ \sigma (H_{2i-1}) \quad i \geq 1 \quad (\checkmark \checkmark) \]

Pf: Let \( H_{2i+1} \) be the optimal solution for 2\( i+1 \) disks. Then it moves disks from peg 1 to peg 2. First optimally move \( 2i \) disks from peg 1 to peg 3 by \( H_{2i} \). Then move the \( (2i+1) \)th disk from peg 1 to peg 2 via peg 3. Then move the \( 2i \) disks from peg 3 to peg 2 via \( \sigma^{-1} (H_{2i}) \).

A similar argument works for \( H_{2i} \).
Now define
\[ \varphi(a) = \overline{a} \quad \varphi(\overline{a}) = ac \]
\[ \varphi(b) = \overline{c} \quad \varphi(\overline{c}) = cb \]
\[ \varphi(c) = \overline{b} \quad \varphi(\overline{b}) = ba \]

**Lemma.** Let \( \Sigma = \{a, b, c, \overline{a}, \overline{b}, \overline{c}\} \).

Then
\[ \varphi(\sigma(x)) = \sigma^{-1}(\varphi(x)) \quad (\star) \]
\[ \varphi(\sigma^{-1}(x)) = \sigma(\varphi(x)) \quad (\star\star) \]

**Pf.** Just verify for each letter.

**Lemma.**
\[ H_{2i+1} = \varphi(H_{2i}) a \quad (\star) \]
\[ H_{2i+2} = \varphi(H_{2i+1}) b \quad (\star\star) \]

for \( i \geq 0 \).

**Pf.** By induction on \( i \). Check for \( i = 0 \).

Then
\[ H_{2i+1} = H_{2i} a \sigma^{-1}(H_{2i}) \quad (\text{by (\star)}) \]
\[ = \varphi(H_{2i-1}) b a \sigma^{-1}(\varphi(H_{2i-1})) b \quad (\text{by induction and (\star\star)}) \]
\[ = \varphi(H_{2i-1}) \varphi(\overline{c}) \varphi(\sigma(H_{2i-1})) a \]
\[ = \varphi(H_{2i-1} \overline{c} \sigma(H_{2i-1})) a \]
\[ = \varphi(H_{2i}) a \quad (\text{by (\star\star)}) \]
And similarly we can verify the inductive hypothesis for the case of \( i \) even.

Putting this all together, we get

\[
H = \Psi(H)
\]

So \( H \) is 2-automatic. It is generated by the automaton below

![Automaton Diagram]

We will eventually see that \( H \) is squarefree; it does not contain 2 consecutive identical blocks.
Closure properties of automatic sequences

What operations can we do to a $k$-automatic sequence and have it remain $k$-automatic? (Theorem 6.8.1)

Easiest: $(U(n))_{n \geq 0} \rightarrow (U(an+b))_{n \geq 0}$, $a, b \geq 0$

**Proof #1:** Using the $k$-kernel. If $a = 0$ then $U(an+b) = U(b)$, which is a constant and hence trivially $k$-automatic. Otherwise $a > 1$.

Let the $k$-kernel be $K_k(U) = \{ (U_i(n)), (U_2(n)), \ldots, (U_r(n)) \}$. Define $S = \{ (U_i(an+b))_{n \geq 0} : 1 \leq i \leq r, \ 0 \leq c < a+b \}$

Claim: The $k$-kernel of $(U(an+b))_{n \geq 0}$ is a subset of $S$.

**Pf.** Let $V(n) = U(an+b)$.

Consider $(V(k^e \cdot n + j))_{n \geq 0}$ for $0 \leq j < k^e$, $e \geq 0$.

Write (by dividing by $k^e$) $ja + b = d \cdot k^e + f$ for $0 \leq f < k^e$, $0 \leq d < a + b$

Then $V(k^e \cdot n + j) = U(a(k^e \cdot n + j) + b)$

$= U(k^e(an + d) + f)$
Now $(U(k^e \cdot m + t))_{n \geq 0}$ is an element of $V_k(U)$, say $(U_i(m))_{n \geq 0}$ for some $i$. So, substituting $m = an + d$, we get

$$V(k^e \cdot n + t) = V(k^e (an + d) + t) = V_i(an + d)$$

$\forall n \geq 0$.

Hence $(V(k^e \cdot n + t))_{n \geq 0} = (V_i(an + d))_{n \geq 0} \in S$.

Now $|S|$ is finite, and $V_k(U) \subseteq S$, so $V$ is $k$-automatic.

**Proof #2:**

Using a transducer.
A general technique: using a finite-state transducer. A transducer is a generalized nondeterministic automaton with inputs and outputs on every transition. The inputs and outputs can be arbitrary words. Only computations that end in a final state give outputs.

Example: A finite state transducer that maps \((3n)_2\) to \((n)_2\) but \((3n+1)_2\) and \((3n+2)_2\) to \(\emptyset\). We need to keep track of "carries" unaccounted for in long division by 3, as well as whether we have output a 1 yet (to avoid the leading-zeros problem). A state is \(q_i\) or \(q_i'\) where \(i\) is the unaccounted carry and \(1\) indicates we have output a 1. An input of 0 must be treated differently.
Theorem. If $L$ is a regular language and $T$ is a finite-state transducer, then $T(L)$ is regular.

We omit the proof (it can be found in the textbook), more or less, or in the literature under the name "Nivat's Theorem".

Now let us prove that if $(U(n))_{n \geq 0}$ is 2-automatic, so is $(U(3n))_{n \geq 0}$.

Proof. Since $(U(n))_{n \geq 0}$ is 2-automatic, its fibers $I_d = \{ (n)_2 : U(n) = d \}$ are regular. For each fiber $I_d$, apply the transducer of the previous page to it. We get $T(I_d) = \{ (n)_2 : U(3n) = d \}$.

So putting the fibers back together as a DFAO, we get a DFAO generating $(U(3n))_{n \geq 0}$.
A converse to Theorem 6.8.1.

Thm. Let \( a \geq 1 \) be an integer, let \((U(n))_{n \geq 0}\) be a sequence s.t. \((U(anti))_{n \geq 0}\) is \(k\)-automatic for \(0 \leq i < a\). Then \((U(n))_{n \geq 0}\) is \(k\)-automatic.

Pf. Define \( t_i(n) = U(anti), \ 0 \leq i < a, \ n \geq 0 \).

Then we know each of the fibers
\[
\{(n)_k : t_i(n) = d\}, \ d \in \Delta
\]
is regular for all \( d, \ 0 \leq i < a \).

Construct a finite-state transducer mapping
\((n)_k\) to \((anti)_k\); this can be done by implementing the ordinary multiplication algorithm maintaining carries in the states. It follows that each
\[
X_{i,d} = \{ (anti)_k : t_i(n) = d \}
\]
is regular.

Now let
\[
Y_d = \bigcup_{d \in \Delta} X_{i,d}
\]
It is regular and
\[
Y_d = \{ (n)_k : U(n) = d \}
\]
So by our result about fibers, \((U(n))_{n \geq 0}\) is \(k\)-automatic.
Finally, one of the deepest results about automatic sequences is you can transduce the sequence itself and still get an automatic sequence, if the transducer obeys certain properties.

Let $T$ be a deterministic transducer where each letter gets mapped to $t$ letters for some fixed $t \geq 1$. Then if $u = (u(n))_{n \geq 0}$ is $k$-automatic, so is $T(u)$.

For the proof, see Section 6.9 of the text.

**Corollary.**

If $(u(n))_{n \geq 0}$ is a $k$-automatic sequence then $\forall d \in \{0, 1, \ldots, k-1\} \forall n \geq 0$ (\(\sum_{0 \leq i < n} u_i \mod d\))^n_{n \geq 0}

The requirement that the transducer be uniform cannot be relaxed, in general, as the following example shows. We can even take a non-uniform morphism.

Consider the $2$-automatic sequence $a = (a_n)_{n \geq 0}$ where $a_n = \begin{cases} 1, & \text{if } n \text{ is a power of } 2; \\ 0, & \text{otherwise.} \end{cases}$
Consider the morphism $f : 1 \to 10$
\[
0 \to 0
\]
Apply $h$ to $a$; we get
\[
\begin{array}{c}
0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
\downarrow h
\end{array}
\]
\[
0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
\]
\[
\begin{array}{c}
\downarrow
\end{array}
\]

Let $h = 0 \ 1 \ 0 \ 1 \ 0 \ 2 \ 1 \ 6^4 \ 1 \ 0^8 \cdots$

We claim $h$ is not 2-automatic.

To prove this, we use one of the fundamental (but easy) results about regular languages: the pumping lemma.

**Pumping lemma for regular languages:**

Let $L$ be regular. Then $\exists$ a constant $n$ (depending on $L$) s.t. $\forall z \in L, |z| \geq n$, $z$ a factorization $z = uvw$, $|uv| \leq n, |v| \geq 1$ s.t. $uv^i w e L \forall i \geq 0$.
PF.

Let \( Z \in L, |Z| \geq n, \) \( n = \# \) of states in a DFA \( M \) recognizing \( L \). Consider the states encountered on reading prefixes of \( Z \):

\[
q_0, \delta(q_0, Z[1]), \delta(q_0, Z[1..2]), \ldots, \delta(q_0, Z[1..n])
\]

There are \( n+1 \) states, so some state is repeated. Hence the acceptance path looks like

\[
\begin{array}{c}
q_0 \rightarrow p \rightarrow o \\
\end{array}
\]

with \( |uv| \leq n \) and \( |v| \geq 1 \). So \( uv^cw \in L \text{ for } c \geq 0 \).

Now let us apply the pumping lemma to our problem.

Assume \( b \) is 2-automatic. Then the fiber

\[
I_1(b) = \{ (2^r + r)_2 : r \geq 0 \}
\]

is a regular language. Now

\[
(2^r + r)_2 = 1 \overbrace{00 \ldots 0}^{r - \| \log_2 r \| - 1} (r)_2
\]

because

\[
| (r)_2 | = \| \log_2 r \| + 1.
\]

If this is regular, so is

\[
A = \{ 0^{r - \| \log_2 r \| - 1} (r)_2 : r \geq 1 \}
\]

which we get from removing the first 1 from each element of \( I_1(b) \).
Apply the pumping lemma to
\[ z = 0^{r - \log_2 r - 1}(r)_2 \]
for \( r \) sufficiently large. Then \( U = 0^k, V = 0^l, W = 0^m(r)_2 \).

Pumping with \( i = 2 \) gives a larger number of 0's at the front, but does not change the \( (r)_2 \) at the end, so
\[ UV^2W \notin A, \]
a contradiction. So \( A \) is not regular and hence \( b \) is not 2-automatic.

---

**Change of base**

**Theorem.** A sequence is \( k^2 \)-automatic if and only if it is \( k^j \)-automatic.

**Proof.** It suffices to prove this for \( j = 1 \).

If \( \hat{a} = (a_n)_{n \geq 0} \) is \( k^i \)-automatic, then each of the fibers
\[ \text{Id} = \{ (n)_{ki} : a_n = d \} \]
is regular. Now apply the morphism sending \( C \)
\[ 0 \leq c \leq k^i - 1, \text{ to } w, \text{ where } |w| = i \text{ and } [w]_k = c, \text{ to Id}. \]
We claim that
\[ \text{rlz} (\psi (\text{Id})) = I' \]
where
\[ I' = \{ (n)_k : a_n = d \} \]

Since regular languages are closed under
1) morphism
2) rlz
the result is regular. So \( A \) is \( k \)-automatic.

For the other direction, assume \( A \) is \( k \)-automatic.

Then by Cobham's little theorem (Lecture 3, p. 8)

\( \exists \) a \( k \)-uniform morphism \( h \) and a coding \( \tau \) and a letter \( c \) s.t. \( a = \tau (h^\omega (c)) \).

Then \( h^\omega (c) = g^\omega (c) \) where \( g = h^i \). So \( a = \tau (g^\omega (c)) \) which again by Cobham's little theorem shows that \( A \) is \( k \)-automatic.