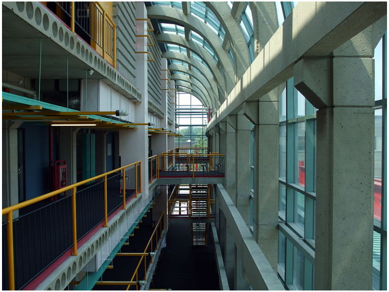
Using Automata to Prove Theorems about Sequences

Jeffrey Shallit

(Joint work with Benoit Cloitre)

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



A new use for automata theory

Everybody here knows about using automata for

- pattern-matching
- lexical analysis
- analysis of finite-state systems
- etc.

In this talk, I will discussing using automata in a new way: to discover and *rigorously* prove certain kinds of theorems in number theory, discrete mathematics, and combinatorics on words.

Walnut

The basic idea:

- We can prove results about $\mathbb{N}$, the natural numbers.
- State the result you want to prove in first-order logic
- *Compile* the first-order logic formula into an automaton accepting the representation of those natural numbers n making the formula true
- Deduce the answer by examining the automaton.

We use a free software package called `Walnut` to do this.

It uses an extension of Presburger arithmetic called *Büchi arithmetic*.

`Walnut` has been used in over 80 papers published in the peer-reviewed literature so far. See
https://cs.uwaterloo.ca/~shallit/walnut.html.

What can you do with `Walnut`?

People have used Walnut to

- find new, conceptually simple proofs of results for which previously only a long, case-based proof was known;
- find and prove entirely new results;
- improve existing results;
- find counterexamples to published claims;
- resolve previously-unsolved conjectures;
- find counterexamples to conjectures.

Find new, conceptually simple proofs of results for which previously only a long, case-based proof was known

Example: Thue's 1912 result on overlap-free sequences.

An *overlap* is a word of the form *axaxa*, where *a* is a single symbol and *x* is a (possibly empty) block.

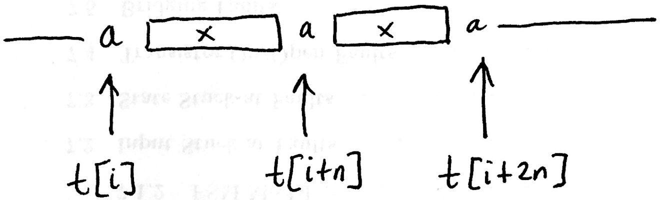
Thue proved that the Thue-Morse word

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

the fixed point of $0 \to 01$ and $1 \to 10$, is overlap-free.

Find new, conceptually simple proofs of results for which previously only a long, case-based proof was known

If $\mathbf{t}$ has an overlap $axaxa$, then it must begin at some position i and we must have $|ax| = n$ for some $n \geq 1$:



So an overlap in $\mathbf{t}$ means there are i, n such that

$$(n \geq 1) \text{ and } \mathbf{t}[i..i+n] = \mathbf{t}[i+n..i+2n]$$

or in other words

$$\exists i, n\ (n \geq 1)\ \wedge\ \forall s\ (0 \leq s \leq n) \implies \mathbf{t}[i+s] = \mathbf{t}[i+s+n].$$

Find new, conceptually simple proofs of results for which previously only a long, case-based proof was known

This logical formula asserts the existence of an overlap in **t**:

$$\exists i, n \ (n \geq 1) \ \land \ \forall s \ (0 \leq s \leq n) \implies \mathbf{t}[i+s] = \mathbf{t}[i+s+n].$$

This formula can be translated into `Walnut` as follows:

```
[Walnut]$ eval hasolap "Ei,n (n>=1) & As (s<=n)
   => T[i+s]=T[i+s+n]";
computed ~:1 states - 35ms
computed ~:2 states - 2ms

-----
FALSE
```

and `Walnut` returns `FALSE`. So there is no overlap.

Walnut syntax explained

```
eval hasolap "Ei,n (n>=1) & As (s<=n)
   => T[i+s]=T[i+s+n]";
```

- `def` defines an automaton for future use
- `eval` determines if formula with no free variables is `TRUE` or `FALSE`
- `E` is an abbreviation for ∃, “there exists”
- `A` is an abbreviation for ∀, “for all”
- `&` is logical `AND`
- `=>` is logical implication
- `~` is logical `NOT`
- `T` is `Walnut`’s way of writing the Thue-Morse sequence

Find and prove entirely new results

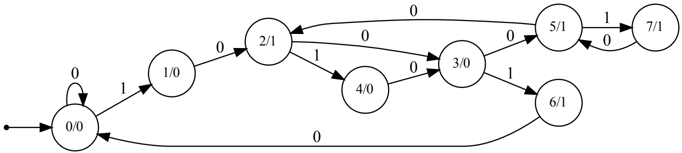
Example: Avoidance of xxx^R.

Can one construct an aperiodic infinite binary word with no instances of the pattern xxx^R?

Idea: guess that there is an automatic sequence generated by a "small" automaton with the desired property, search for it with breadth-first search, and then verify it with Walnut.

A breadth-first search quickly finds a candidate automaton `FB` with 8 states.

Find and prove entirely new results



Then we can verify this automaton FB generates a sequence 001001101 $\cdots$ with the desired property with Walnut as follows:

```
eval claim1 "?msd_fib ~Ei,p p>0 & At (t>i) => FB[t]=FB[t+p]":
eval claim2 "?msd_fib ~Ei,n n>0 & At (t<n) => (FB[i+t]=FB[i+n+t]
   & FB[i+t]=FB[(i+3*n)-(t+1)])":
```

Improve existing results

Example: unbordered factors of the Thue-Morse word $\mathbf{t}$ and the Currie-Saari result.

A word w is said to be *bordered* if there exist words x, y with x nonempty such that $w = xyx$. Otherwise it is *unbordered*.

Currie and Saari were interested in the lengths of unbordered factors of the Thue-Morse word $\mathbf{t}$.

They proved: *an unbordered factor exists provided $n \not\equiv 1 \pmod 6$.*

However, this criterion is sufficient but not necessary: 0011010010110100110010110100101 is a factor of length 31 that is unbordered.

We can ask `Walnut` to create an automaton for the lengths for which unbordered factors exist.

Improve existing results

```
def tmfactoreq "At t<n => T[i+t]=T[j+t]":
```

Given i, j, n, assert that the length-n factors beginning at position i and j of $\mathbf{t}$ are the same.

```
def tmbord "j>=1 & j<n & $tmfactoreq(i,(i+n)-j,j)":
```

Given i, j, n, assert that the length-n factor beginning at position i has a border of length j.

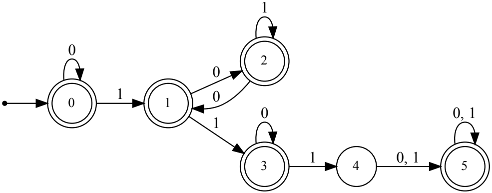
```
def tmunblength "Ei Aj ~$tmbord(i,j,n)":
```

Given n, assert that there is some length-n factor having no borders of any length.

Improve existing results

```
def tmfactoreq "At t<n => T[i+t]=T[j+t]":
def tmbord "j>=1 & j<n & $tmfactoreq(i,(i+n)-j,j)":
def tmunblength "Ei Aj ~$tmbord(i,j,n)":
```

This generates the following automaton:



So we have proved a necessary and sufficient condition:

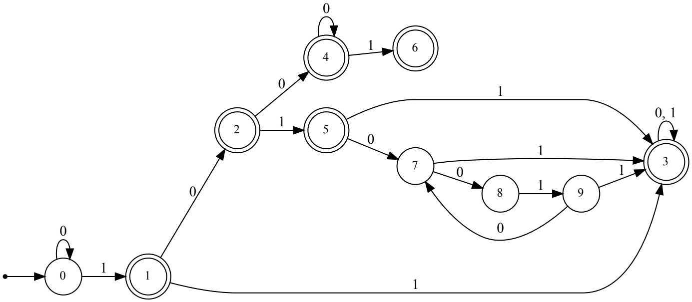
Theorem. The Thue-Morse word $\mathbf{t}$ has an unbordered factor of length n if and only if $(n)_2 \not\in 1(01^*0)^*10*1$.

Find counterexamples to published claims

A paper once claimed that "Every length-k factor of the Thue-Morse word $\mathbf{t}$ appears as a factor of every length-$(8k-1)$ factor of $\mathbf{t}$."

This claim is false in general. Let's determine those k for which it is true.

```
def al "Ai,j El (l>=j) & (l+1<=j+7*k) & As (s<k) => T[i+s]=T[l+s]":
```



Resolve previously-unsolved conjectures

Example: Rampersad's conjecture on generalized paperfolding sequences

A paperfolding sequence $\mathbf{P_f}$ is an infinite binary sequence $p_1 p_2 p_3 \cdots$ specified by an infinite sequence of binary unfolding instructions $f_0 f_1 f_2 \cdots$, as the limit of the infinite words $\mathbf{P}_{f_0 f_1 f_2 \cdots}$, defined as follows:

$$\mathbf{P}_\varepsilon = \varepsilon;$$

$$\mathbf{P}_{f_0 \cdots f_{i+1}} = \mathbf{P}_{f_0 \cdots f_i} \; f_{i+1} \; \overline{\mathbf{P}^R_{f_0 \cdots f_i}}.$$

For example, if $\mathbf{f} = 000 \cdots$, we get the simplest paperfolding sequence

$$\mathbf{p} = 00100110001101100010011100011011 \cdots .$$

Resolve previously-unsolved conjectures

Narad Rampersad once conjectured that if $\mathbf{f}$ and $\mathbf{g}$ are two distinct infinite sequences of unfolding instructions, then the paperfolding sequences $\mathbf{P_f}$ and $\mathbf{P_g}$ have only finitely many common factors.

Theorem

For all finite sequences of unfolding instructions f and g, if f differs from g in the k'th position, then $\mathbf{P_f}$ and $\mathbf{P_g}$ have no factors of length $14 \cdot 2^k$ in common.

We can prove this with `Walnut`, but it takes a bit of work.

The basic idea (due to Luke Schaeffer) is to find a *single* finite automaton that encodes *all* the *uncountably many* paperfolding sequences simultaneously.

Find counterexamples to conjectures

Let $r(k, A, n)$ denote the number of representations of n as a sum of k elements of a set $A \subseteq \mathbb{N}$.

In 2002, Dombi conjectured that if A is co-infinite, then the sequence $(r(3, A, n))_{n \geq 0}$ cannot be strictly increasing.

Using `Walnut`, we gave an explicit counterexample where $\mathbb{N} \setminus A$ is co-infinite, and even has positive lower density, but $(r(3, A, n))_{n \geq 0}$ is strictly increasing.

Find counterexamples to conjectures

Sketch of proof: Let $F = \{3, 12, 13, 14, 15, 48, 49, 50, \ldots\}$ be the set of natural numbers whose base-2 expansion (ignoring leading zeros) is of even length and begins with 11.

Set $A = \mathbb{N} \setminus F$.

Using Walnut, find a linear representation for $d(n)$, the first difference of the number of representations as sum of 3 elements of A. We want $d(n) > 0$.

Then we show $f(n) := d(n) - 4d(\lfloor n/4 \rfloor)$ is an automatic sequence, and we can explicitly determine the automaton for it.

This automaton gives the inequality

$$d(n) \geq 4d(\lfloor n/4 \rfloor) - 18,$$

which is enough to show by induction that $d(n) > 0$ for all n.

How does `Walnut` work?

- The logical formula is parsed and compiled into a deterministic finite automaton.
- The automaton has the property that it accepts exactly the values of the free variables (in parallel) that make the formula true.
- Addition is performed with an automaton with three inputs that verifies the relation $x + y = z$. Easy in base b, harder for Zeckendorf representation.
- $\exists$ is achieved by projection of the transitions corresponding to the named variables. A transition on $[x_i, y_i]$ becomes a transition on y_i after applying $\exists x$. This can result in an NFA, so the automaton is determinized and minimized.
- $\forall$ is achieved by using de Morgan's law.
- Worst-case running time is a tower of exponentials corresponding to number of quantifier alternations.

Cloitre's sequence $a(n)$

Invented by Benoit Cloitre in May 2005.

Let $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ be the Fibonacci numbers.

Define

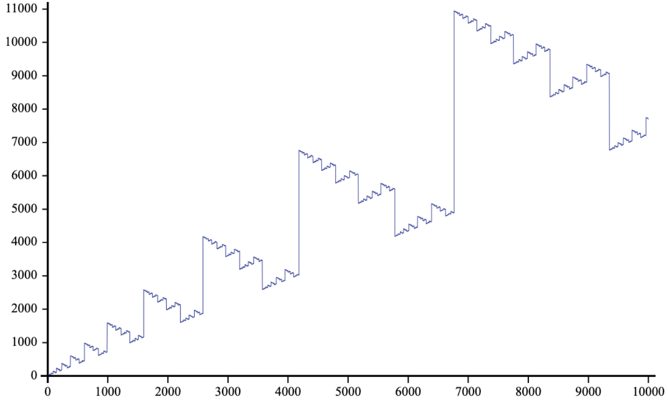
$$a(n) = \begin{cases} n, & \text{if } n \leq 1; \\ F_{j+1} - a(n - F_j), & \text{if } F_j < n \leq F_{j+1} \text{ for } j \geq 2. \end{cases}$$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a(n)$	0	1	1	2	4	4	7	7	6	12	12	11	9	9	20	20

It is sequence A105774 in the OEIS (On-Line Encyclopedia of Integer Sequences).

The graph of Cloitre's sequences $a(n)$

The sequence has an intricate fractal structure:



Cloitre's sequence $a(n)$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a(n)$	0	1	1	2	4	4	7	7	6	12	12	11	9	9	20	20

The kinds of things we might want to know include

- Which integers do not appear in it?
- How often does each integer appear in it?
- Which integers appear only once?
- What are upper and lower bounds on the growth rate of $a(n)$?
- When do consecutive equal terms appear?
- What are values at special indices, like F_n?
- What about the sequence arising by sorting the terms in ascending order?

Believe it or not, we can answer these questions using automata theory!

Fibonacci (Zeckendorf) representation

- The Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

- In analogy with base-2 representation, we can represent every non-negative integer n in the form

$$n = \sum_{0 \le i \le t} \epsilon_i F_{i+2} \quad \text{with} \quad \epsilon_i \in \{0, 1\}.$$

Fibonacci (Zeckendorf) representation

- But then some integers have multiple representations, e.g., $14 = 13 + 1 = 8 + 5 + 1 = 8 + 3 + 2 + 1$
- So to get uniqueness of the representation, we impose the additional condition that $\epsilon_i \epsilon_{i+1} = 0$ for all i: never use two adjacent Fibonacci numbers.
- Usually we write the representation in the form

$$(n)_F = \epsilon_t \epsilon_{t-1} \cdots \epsilon_0,$$

with most significant digit first. So, for example, $(19)_F = 101001$. This is called *Zeckendorf representation*.

Édouard Zeckendorf (1901–1983), Belgian amateur mathematician

An automaton for the sequence

Now that we have Zeckendorf representation, we can deal with automata that compute functions of the natural numbers: the inputs to the automata are Zeckendorf representations of $\mathbb{N}$.

Amazing thing: there is a finite automaton that computes $a(n)$ in the following sense: it takes the Zeckendorf representations of n and x as inputs, in parallel, and accepts if and only if $x = a(n)$.

(We might have to pad the shorter with leading zeroes, to make the representations of n and $a(n)$ the same length.)

An automaton for the sequence

Here it is:

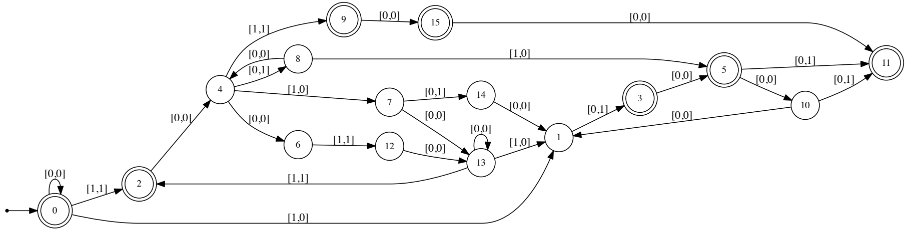


Figure 1: Automaton computing $a(n)$.

Example: $a(15) = 20$, $(15)_F = 100010$, $(20)_F = 101010$, and the automaton accepts $[1,1][0,0][0,1][0,0][1,1][0,0]$.

How did we find the automaton?

We *guessed* the automaton using a version of the Myhill-Nerode theorem, as follows:

We *guess* that $\{[0,0]^*(n,x)_F \,:\, x = a(n)\}$ is regular.

The Myhill-Nerode theorem tells us that each state of the minimal automaton for a regular language L corresponds to the language $L_x = \{y \,:\, xy \in L\}$.

Of course we cannot compute L_x from empirical data alone, but we can compute sets like $L_{x,c} = \{y \,:\, |y| \leq c \text{ and } xy \in L\}$.

How did we find the automaton?

If we assume that (say) $L_x = L_y$ if and only if $L_{x,c} = L_{y,c}$ for some small integer c, we can guess the automaton.

We can compute the number of states needed for $c = 1, 2, 3, \ldots$ until this number stabilizes.

This gives a conjectured automaton A for L.

But it is just a guess...so far.

How did we find the automaton?

How can we verify that our guessed automaton is correct?

First step: we need to verify that A really computes a function, that is, for each n there is exactly one x such that (n, x) is accepted.

Then we need to verify that the function it computes obeys the defining recurrence: $a(n) = F_{j+1} - a(n - F_j)$ if $F_j < n \leq F_{j+1}$ for $j \geq 2$.

Both of these claims can be phrased in first-order logic.

For example, to say that an automaton $a(n, x)$ computes a function means

$$\forall n \; \exists x \; a(n, x)$$

and

$$\neg \exists n, x, y \; x \neq y \; \wedge \; a(n, x) \; \wedge \; a(n, y).$$

How did we verify the automaton?

Let's check *a* computes a function:

```
eval check_at_least_one "?msd_fib An Ex $a(n,x)":
eval check_at_most_one "?msd_fib ~En,x,y x!=y & $a(n,x) &
   $a(n,y)":
```

and `Walnut` returns `TRUE` for both assertions.

Here `?msd_fib` is a bit of jargon saying that all numbers are expressed in Zeckendorf representation.

How did we verify the automaton?

Next we must verify that our automaton obeys the defining recurrence $a(n) = F_{j+1} - a(n - F_j)$ if $F_j < n \leq F_{j+1}$ for $j \geq 2$.

```
reg adjfib msd_fib msd_fib "[0,0]*[0,1][1,0][0,0]*":
# accepts (F_k, F_{k+1})
def trapfib "?msd_fib $adjfib(x,y) & x<k & y>=k":
# accepts (k,x,y) if x is the largest Fibonacci number
# less than k and y is the next largest Fib number
eval test105774 "?msd_fib Ak,x,y,z,t ($trapfib(k,x,y)
   & $a(k,z) & $a(k-x,t)) => y=z+t":
```

and `Walnut` returns `TRUE`. At this point we know that our guessed automaton is correct.

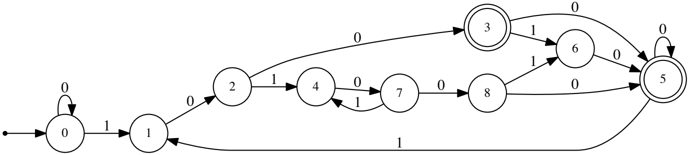
Which integers don't appear in $(a(n))$?

They are

$$3, 5, 8, 10, 13, 16, 18, 21, 24, 26, 29, 31, 34, 37, 39, 42, \ldots$$

```
def dont_appear "?msd_fib ~Ek $a(k,n)":
```

And this gives the automaton below.



Which integers don't appear in $(a(n))$?

Do you recognize those numbers

$$3, 5, 8, 10, 13, 16, 18, 21, 24, 26, 29, 31, 34, 37, 39, 42, \ldots \ ?$$

No? Then look them up in the OEIS:

Search: **seq:3,5,8,10,13,16,18,21,24,26,29,31,34,37**

Displaying 1-1 of 1 result found. page 1

Sort: relevance | references | number | modified | created Format: long | short | data

A004937 a(n) = round(n*phi^2), where phi is the golden ratio, A001622. +30 4

0, **3, 5, 8, 10, 13, 16, 18, 21, 24, 26, 29, 31, 34, 37**, 39, 42, 45, 47, 50, 52, 55, 58, 60, 63, 65, 68, 71, 73, 76, 79, 81, 84, 86, 89, 92, 94, 97, 99, 102, 105, 107, 110, 113, 115, 118, 120, 123, 126, 128, 131, 134, 136, 139, 141, 144, 147, 149, 152, 154, 157

(list; graph; refs; listen; history; text; internal format)

Which integers don't appear in $(a(n))$?

Now

$$\begin{aligned} \mathrm{rnd}(n \cdot \varphi^2) &= \lfloor \varphi^2 n + 1/2 \rfloor \\ &= \lfloor (\varphi^2 2n + 1)/2 \rfloor \\ &= \lfloor (\lfloor \varphi^2 2n \rfloor + 1)/2 \rfloor. \end{aligned}$$

So we can use the following Walnut code to verify our guess:

```
def a004937 "?msd_fib En,x $phi2n(2*n,x) & z=(x+1)/2":
eval check_dont "?msd_fib An (n>0) =>
   ($a004937(n) <=> (~Ek k>0 & $a(k,n)))":
```

and Walnut returns TRUE.

Elements appear at most twice

Proposition

No natural number appears three or more times in A105774.

Proof.

We use the following `Walnut` code.

```
eval test012 "?msd_fib ~Ex,y,z,n x<y & y<z & $a(x,n) &
   $a(y,n) & $a(z,n)":
```

and `Walnut` returns `TRUE`. □

Elements appearing twice

Proposition

If a number appears twice in $(a(n))_{n\geq 0}$, the two occurrences are consecutive.

Proof.

We use the following `Walnut` code:

```
eval twice_consec "?msd_fib An,x,y (x<y & $a(x,n) & $a(y,n))
   => y=x+1":
```

and `Walnut` returns `TRUE`. □

Fixed points

Proposition

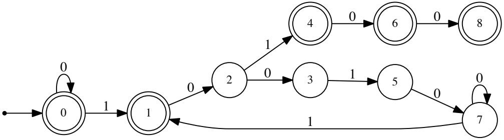
We have $a(n) = n$ *for* $n > 0$ *if and only if* $(n)_F \in 1(00100^*1)^*\{\epsilon, 01, 010, 0100\}$.

Proof.

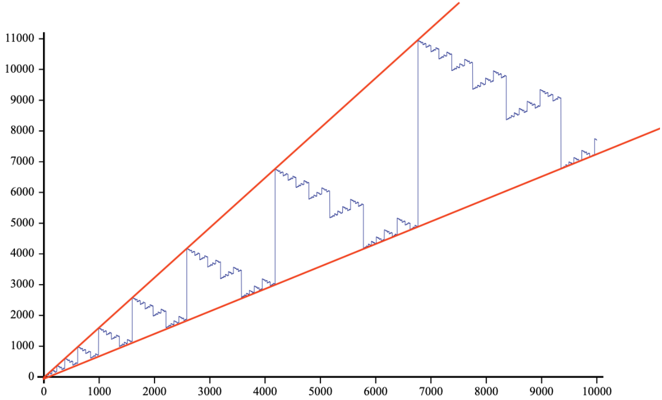
We use the `Walnut` command

```
def fixed "?msd_fib $a(n,n)":
```

and it produces the automaton below, from which we can directly read off the result.



Upper and lower bounds



The function $a(n)$ seems very tightly bounded, above and below, by lines $\beta_1 n$ and $\beta_2 n$.

Upper and lower bounds

Numerical experiments suggest the following result:

Proposition

For all $n \geq 0$ we have $\lfloor \frac{\varphi+2}{5} n \rfloor \leq a(n) \leq \lfloor \varphi n \rfloor$.

We can prove this with the following `Walnut` code:

```
eval lowerbound "?msd_fib An,x,y ($a(n,x) & $phin(n,y))
   => x>=(y+2*n)/5":
eval upperbound "?msd_fib An,x,y ($a(n,x) & $phin(n,y))
   => x<=y":
```

Upper and lower bounds

Now we need to show these bounds are tight. More precisely:

Proposition

We have $\liminf_{n\to\infty} a(n)/n = \frac{\varphi+2}{5}$ *and* $\limsup_{n\to\infty} a(n)/n = \varphi$.

Proving this requires a bit more cleverness, because the bounds are only approached rarely.

Upper and lower bounds

Recall the Lucas numbers: $L_0 = 2$, $L_1 = 1$, $L_n = L_{n-1} + L_{n-2}$.

We have $L_n = F_{n-1} + F_{n+1}$, so the Zeckendorf representation of L_n is 1010^{n-3}.

For the claim $\liminf_{n\to\infty} a(n)/n = \frac{\varphi+2}{5}$, using the well-known Binet formulas for the Fibonacci and Lucas numbers, it suffices to show that $a(L_k + 1) = F_{k+1} + 1$ for all $k \geq 3$.

```
reg lucfib msd_fib msd_fib "[0,0]*[1,1][0,0][1,0][0,0]*":
# regular expression for the pair (L_k, F_{k+1}) for k>=3
eval chklow "?msd_fib Ax,y $lucfib(x,y) => $a(x+1,y+1)":
```

Upper and lower bounds

For the claim $\limsup_{n\to\infty} a(n)/n = \varphi$ it suffices to show that $a(F_k + 1) = F_{k+1} - 1$ for all $k \geq 2$.

This follows directly from the defining recurrence for $a(n)$.

Or one can use Walnut:

```
eval chkup "?msd_fib Ax,y,m ($adjfib(x,y) & $a(x+1,m))
   => m+1=y":
```

More results on Cloitre's sequence

Many, many more results about Cloitre's sequence can be proved using `Walnut`.

See https://arxiv.org/abs/2312.11706 for more of them.

Conclusions

- Automata provide a *new tool* for solving certain kinds of problems number theory and combinatorics, and can give *rigorous proofs*.
- The method cannot deal with all sequences, but only sequences generated with automata.
- To be amenable, the problem must have a close relationship with some system of numeration, such as base 2 or Zeckendorf representation.
- Guessing the automaton and then checking it satisfies a definition often works in practice.
- The worst-case running time of deciding the needed formulas can be truly astonishingly large, but in many cases terminates quickly.

The Walnut Prover

Our publicly-available prover, originally written by Hamoon Mousavi, is called `Walnut` and can be downloaded from

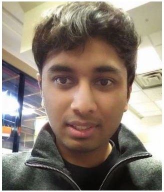
https://cs.uwaterloo.ca/~shallit/walnut.html .

Lagniappe

There is a finite automaton of 97 states, that on input 10^n in Zeckendorf representation, outputs the n'th decimal digit of $\varphi = (1 + \sqrt{5})/2$!

Designer and Implementers of Walnut

Hamoon Mousavi—Designer and Implementer



Aseem Baranwal—implementer

Laindon C. Burnett—implementer

Anatoly Zavyalov—implementer

For further reading

Available at
a fine bookstore
near you!