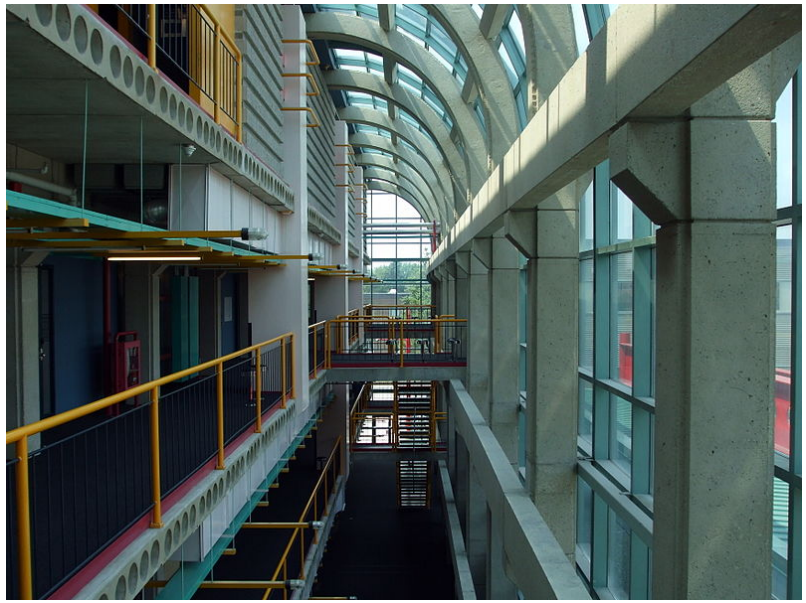
Using Walnut to Prove Results About Sequences in the OEIS

Jeffrey Shallit

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



The OEIS

The *On-Line Encyclopedia of Integer Sequences* (OEIS) is an enormous database of mathematical information, containing over 357,000 integer sequences and theorems, conjectures, and citations to papers about them.

We owe Neil Sloane a huge debt of gratitude for his work on this.

And also to all the volunteers who edit the database!

10^6 thanks to everyone!

This talk

What I will do in this talk:

- discuss a theorem prover called `Walnut` that can "automatically" prove many results about sequences in the OEIS
- illustrate its use in proving some theorems
- explain how you can use it in your own work
- talk about its limitations

What is Walnut?

- Free software, written in Java.
- Originally designed by Hamoon Mousavi.
- Additions and changes by Aseem Raj Baranwal, Laindon C. Burnett, Kai Hsiang Yang, and Anatoly Zavyalov.
- Available at https://cs.uwaterloo.ca/~shallit/walnut.html.
- Rigorously proves theorems about the natural numbers and sequences.
- Has been used in 60 papers in the literature, to prove dozens of theorems (and even correct some incorrect ones in the literature!)

What can Walnut do?

It can **rigorously prove** theorems about sequences.

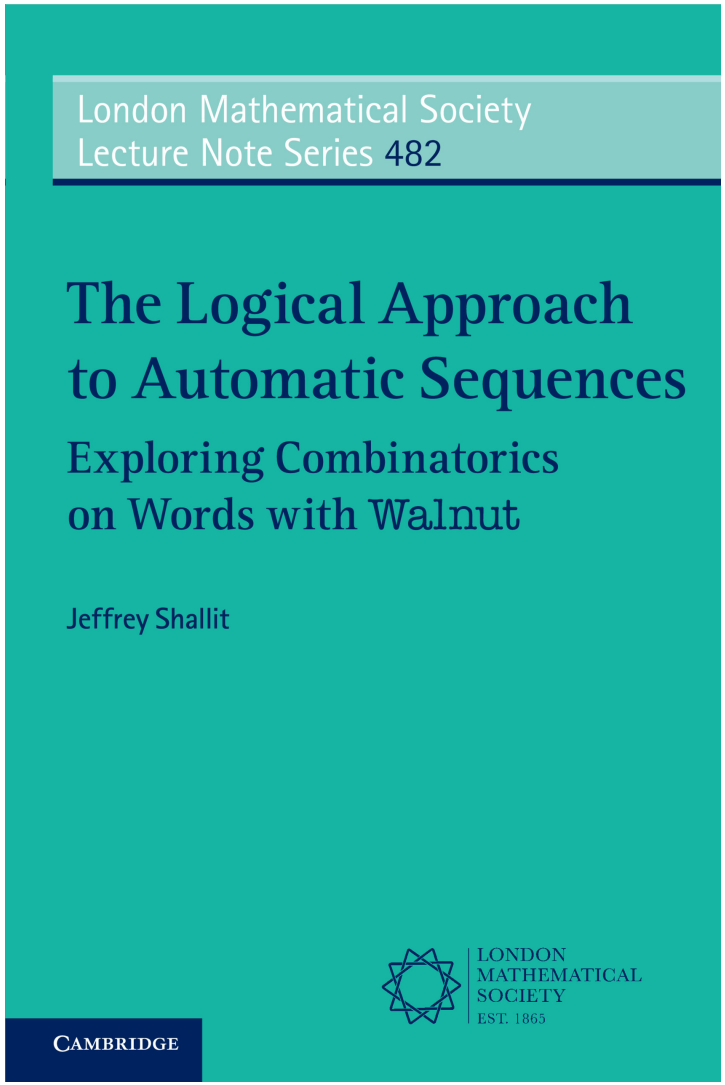
- *But not all sequences!* Just a special class called (generalized) automatic sequences.
  - Examples of sequences in this class include the Thue-Morse sequence, the Rudin-Shapiro sequence, the infinite Fibonacci word, the infinite Tribonacci word, Sturmian words, paperfolding words, etc.
- *But not all theorems!* You have to state the theorem in first-order logic, and you can only do things like add, subtract, and compare integers, and index into the sequence.
  - You can also use the existential ($\exists$) and universal ($\forall$) quantifiers.
  - However, you *can't* do multiplication by variables, or division, square root, arbitrary real numbers, primes, etc.
  - You can multiply or divide by a constant, however.

Other limitations

The running time and space requirements of `Walnut` in the worst-case are extraordinarily high, so sometimes `Walnut` proofs fail because it runs out of space or would take years to complete the proof.

Even so, you can do a lot with it.

Self-promotion: I wrote a book, entitled *The Logical Approach to Automatic Sequences: Exploring Combinatorics on Words with* `Walnut`, which has just been published by Cambridge University Press.



A very simple example: odd plus odd gives even

Let's use `Walnut` to prove this theorem:

Theorem. *The sum of two odd natural numbers is even.*

The first thing you need to do is to translate the theorem into a more precise formulation in the language of first-order logic.

So we will need to define what it means to be "odd" and "even".

A very simple example: odd plus odd gives even

Here are those definitions:

$$\text{odd}(n) := \exists k \ n = 2k + 1$$
$$\text{even}(n) := \exists k \ n = 2k.$$

Here $\exists$ is the symbol for "there exists".

Next, we restate the desired theorem in first-order logic:

$$\forall m, n \ (\text{odd}(m) \wedge \text{odd}(n)) \implies \text{even}(m + n).$$

Here $\forall$ is the symbol for "for all", $\wedge$ is the symbol for "and", $\implies$ is the symbol for implication.

Now we simply translate these into a form `Walnut` can understand.

A very simple example: odd plus odd gives even

```
[Walnut]$ def odd "Ek n=2*k+1";

[Walnut]$ def even "Ek n=2*k";

[Walnut]$ eval thm "Am,n ($odd(m) & $odd(n)) => $even(m+n)":
(odd(m))&odd(n))):2 states - 3ms
 ((odd(m))&odd(n)))=>even((m+n)))):1 states - 2ms
  (A m , n ((odd(m))&odd(n)))=>even((m+n))))):1 states - 1ms
Total computation time: 33ms.

----
TRUE
```

The theorem is now proved.

But the *real* power of Walnut is only apparent when you use it to deal with infinite sequences.

A more serious example

Let's do a more serious example. In preparing for this talk, I searched the OEIS for "Fibonacci conjecture" and I quickly found one that `Walnut` can handle.

A260311 Difference sequence of A260317. 2

1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 3, 2, 3, 2, 3, 3, 2, 3, 3, 2, 3, 5, 3, 2, 3, 5, 3, 2, 3, 5, 3, 5, 3, 2, 3, 5, 3, 5, 3, 2, 3, 5, 3, 5, 5, 3, 5, 3, 2, 3, 5, 3, 5, 5, 3, 5, 3, 2, 3, 5, 3, 5, 5, 3, 5, 3, 5, 5, 3, 5, 3, 2, 3, 5, 3, 5, 5, 3, 5, 3, 5, 5, 3, 5, 3 (list; graph; refs; listen; history; text; internal format)

OFFSET 1,6

COMMENTS Conjecture: a(n) is a Fibonacci number (A000045) for every n.

A260317 Numbers not of the form v(m) + v(n), where v = A001950 (upper Wythoff numbers) and 1 <= m <= n - 1, for n >= 2. 2

1, 2, 3, 4, 5, 6, 8, 10, 11, 13, 14, 16, 19, 21, 24, 26, 29, 32, 34, 37, 40, 42, 45, 50, 53, 55, 58, 63, 66, 68, 71, 76, 79, 84, 87, 89, 92, 97, 100, 105, 108, 110, 113, 118, 121, 126, 131, 134, 139, 142, 144, 147, 152, 155, 160, 165, 168, 173, 176, 178, 181 (list; graph; refs; listen; history; text; internal format)

OFFSET 1,2

COMMENTS It appears that the difference sequence consists entirely of Fibonacci numbers (A000045); see A260311.

A001950 Upper Wythoff sequence (a Beatty sequence): a(n) = floor(n*phi^2), where phi = (1+sqrt(5))/2. (Formerly M1332 N0509) 243

2, 5, 7, 10, 13, 15, 18, 20, 23, 26, 28, 31, 34, 36, 39, 41, 44, 47, 49, 52, 54, 57, 60, 62, 65, 68, 70, 73, 75, 78, 81, 83, 86, 89, 91, 94, 96, 99, 102, 104, 107, 109, 112, 115, 117, 120, 123, 125, 128, 130, 133, 136, 138, 141, 143, 146, 149, 151, 154, 157 (list; graph; refs; listen; history; text; internal format)

Solving the conjecture

First, we need something for the *upper Wythoff sequence:* this is the map

$$n \to \lfloor \varphi^2 n \rfloor,$$

where $\varphi = (1 + \sqrt{5})/2$ is the golden ratio.

Luckily, I already had some Walnut code for this. (I'll explain how I got it, later).

In `Walnut`, though, the only functions that we can handle *directly* have finite range.

So instead we use a subterfuge: we define a function of *two* arguments, n and x, such that the result is `TRUE` if and only if $x = \lfloor \varphi^2 n \rfloor$.

In Walnut the assertion that $x = \lfloor \varphi^2 n \rfloor$ is then expressed as follows:

$$\texttt{phi2n(n, x)}.$$

Solving the conjecture

Next, we need code for the OEIS sequence A260317.

Its description in the OEIS says

"Numbers not of the form $v(m) + v(n)$, where $v =$ A001950 (upper Wythoff numbers) and $1 \leq m \leq n-1$ for $n \geq 2$".

How shall we write this as a first-order statement?

Let's see: we want something that that says $\mathrm{s2uw(z)}$ is true iff z belongs to A260317.

Solving the conjecture

In other words $\mathrm{s2uw(z)}$ is true iff there *do not* exist m, n, x, y such that
$z = x + y$
where `phi2n(m,x)`
and `phi2n(n,y)`
and $1 \leq m$
$m \leq n - 1$
$n \geq 2$.
So we just write this as a first-order statement:

```
def s2uw "?msd_fib ~Em,n,x,y z=x+y & $phi2n(m,x)
   & $phi2n(n,y) & 1<=m & m<=n-1 & n>=2":
```

Here ~ is logical NOT.

Solving the conjecture

Now we need the gaps g between successive values of A260317.

To do that we say that there exist t, v such that
$t < v$
and s2uw(t) holds
and s2uw(v) holds
but for all u between t and v, the assertion s2uw(u) does not hold,
and the gap size $g = v - t$.

```
def gap "?msd_fib Et,v t<v & $s2uw(t) & $s2uw(v) &
     (Au (u>t & u<v) => ~$s2uw(u)) & g=v-t":
```

The result is an assertion gap(g) which is true if and only if g belongs to A260311.

Solving the conjecture

Finally, we assert that every gap is a Fibonacci number:

```
reg isfib msd_fib "0*10*":
eval thm "?msd_fib Ax $gap(x) => $isfib(x)":
```

and here is what we get:

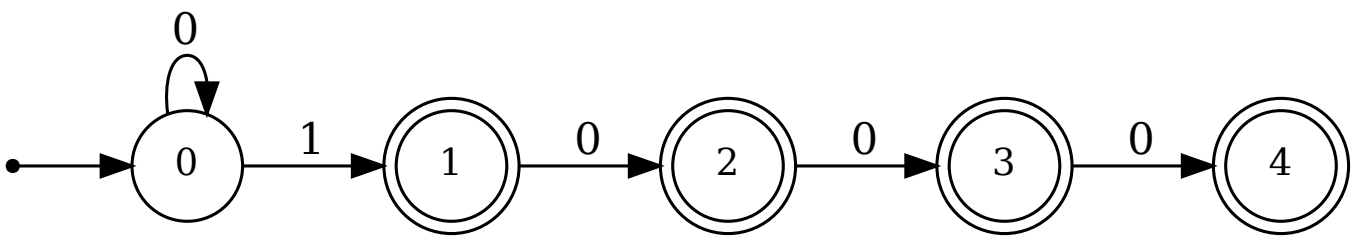
```
[Walnut]$ eval thm "?msd_fib Ax $gap(x) => $isfib(x)":
(gap(x))=>isfib(x))):2 states - 44ms
 (A x (gap(x))=>isfib(x)))):1 states - 11ms
Total computation time: 96ms.

----
TRUE
```

And the theorem is proved!

Going further

In fact, we get even more. How is gap stored? It is a *finite automaton* that accepts the Fibonacci representation of those g that are elements of A260311. Namely:



By examining that automaton, we actually obtain something more:

Theorem. *The only possible gaps are* $1, 2, 3, 5$.

How did we get phi2n?

We obtained the automaton for phi2n using a theorem in a paper of Don Reble in the OEIS!

Theorem. *We have* $\lfloor n\varphi^2 \rfloor = x + 2$, *where* x *is the number obtained by taking the Fibonacci representation of* $n - 1$ *and concatenating two zeros on the end.*

For the proof, see https://oeis.org/A007895/a007895.pdf.

How does it work?

Internally, assertions such as `gap` and `s2uw` are stored as *finite automata*.

A finite automaton is a simple model of a computer. There are two variations that we use: an automaton with output (DFAO), that can compute a function of its input, and an automaton (DFA) as accepter/rejecter of its input.

With each logical formula f, we associate a DFA. The DFA has one or more inputs; these are the variables of the formula. The DFA accepts exactly those natural number values of the variables that make the formula f true.

How does it work?

In automaton diagrams, states are represented by circles or ovals.

A DFA starts in a start state (denoted by a headless arrow coming into the state).

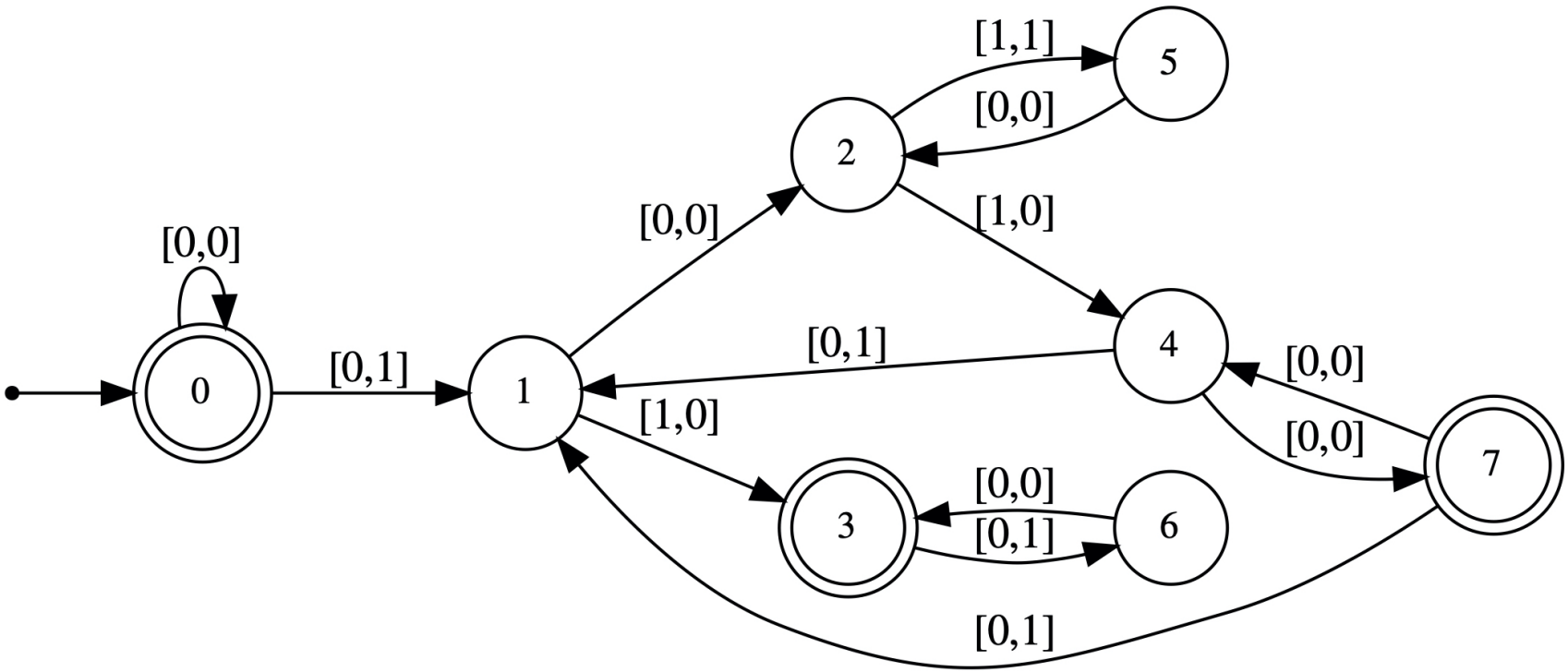
It processes the symbols of the input one-by-one, following the arrow labeled with the symbol.

If, after processing the whole input, it is in a final state (denoted by double circle), the input is accepted. Otherwise it is rejected.

By contrast, a DFAO returns an output specified in the state last reached when processing the input.

The automaton for phi2n

Here is the DFA for phi2n.



For example, phi2n(10,26) is true. Check with input $[0,1][0,0][1,0][0,1][0,0][1,0][0,0]$.

How does it work?

Walnut compiles a logical formula into the appropriate automaton. Each logical and arithmetic operation corresponds to some well-studied automaton transformation that can be carried out.

Some of these operations only increase the automaton size by a small amount. For example, AND and OR only multiply the sizes of the two automata.

Other operations, like $\forall$, can blow up the size of the automata exponentially.

This means that if there are t quantifier alternations, then the resulting automaton could be, in the worst case, of size something like

$$2^{2^{2^{\cdots 2^n}}}.$$

How are numbers represented?

Numbers in `Walnut` are represented in some *numeration system*.

Typically, the numeration system has to be geared to the problem in some way.

`Walnut` can handle

- base-k representation for any fixed $k \geq 2$
- Fibonacci representation (aka Zeckendorf representation), where numbers are represented as sums of Fibonacci numbers
- Tribonacci representation
- Pell representation
- Ostrowski representation
- base-$(-k)$ representation

What kind of sequences can Walnut prove results about?

Walnut can prove first-order logical statements about automatic sequences.

These are sequences that are expressible as the outputs of DFAO's where the input is one of the 6 types of numeration system listed above.

In particular, Walnut can handle words that are images (under a coding) of a fixed-point of a k-uniform morphism.

Another example

Let us use `Walnut` to solve a previously-unsolved problem of Vladimir Shevelev.

He observed that for the Thue-Morse sequence

$$\mathbf{t} = t_0 t_1 t_2 \cdots = 0110100110010110 \cdots$$

there do not exist integers $0 < i < j$ such that

$$t_n \in \{t_{n+i}, t_{n+j}\}$$

for all n.

Here $t(n)$ is the number of 1's, computed mod 2, in the binary representation of n.

We can prove this with `Walnut` as follows:

```
eval shev1 "Ei,j 0<i & i<j & An (T[n]=T[n+i]|T[n]=T[n+j])":
```

Another example

However, there do exist integers $0 < i < j < k$ such that

$$t_n \in \{t_{n+i}, t_{n+j}, t_{n+k}\}$$

for all n.

Shevelev asked to characterize these valid triples (i, j, k).

We can solve this problem by finding an automaton that accepts all valid triples, as follows:

```
def shev2 "0<i & i<j & j<k &
   An (T[n]=T[n+i]|T[n]=T[n+j]|T[n]=T[n+k])":
```

This was a big computation in `Walnut`! It used 6432 seconds of CPU time and 18 Gigs of RAM. The largest intermediate automaton had 2952594 states.

Another example

The resulting automaton `shev2` has 53 states, and encodes all the valid triples (i, j, k).

With it we can easily determine if a given triple has the desired property.

We can also use it to prove various results of Shevelev, such as

Theorem. *All triples of the form* $(a, a + 2^j, a + 2^k)$ *for* $a \geq 1$, $0 \leq j < k$, *are valid.*

Proving conjectures by guessing the automaton

`Walnut` can sometimes *prove* conjectures obtained by *guessing!*

The idea is to "guess" an automaton for a sequence using heuristics.

Once the automaton is guessed, we then *rigorously verify* that it is correct using `Walnut`.

Let us work through an example.

Proving conjectures by guessing the automaton

Consider OEIS sequence A140100:

A140100	Start with Y(0)=0, X(1)=1, Y(1)=2. For n > 1, choose least positive integers Y(n) > X(n) such that neither Y(n) nor X(n) appear in {Y(k), 1 <= k < n} or {X(k), 1 <= k < n} and such that Y(n) - X(n) does not appear in {Y(k) - X(k), 1 <= k < n} or {Y(k) + X(k), 1 <= k < n}; sequence gives X(n) (for Y(n) see A140101).	25

1, 3, 4, 6, 7, 9, 10, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 29, 30, 32, 34, 35, 37, 38, 40, 41, 43, 44, 46, 47, 49, 51, 52, 54, 55, 57, 58, 60, 61, 63, 64, 66, 67, 69, 71, 72, 74, 75, 77, 78, 80, 82, 83, 85, 86, 88, 89, 91, 92, 94, 95, 97, 98, 100, 102, 103, 105, 106

Julien Cassaigne conjectured that for all k the sum $X(k) + Y(k)$ equals either $X(Y(k))$ or $Y(X(k))$.

Our goal is to prove this conjecture.

Proving conjectures by guessing the automaton

The definition of the sequences $X(k)$ and $Y(k)$ are as follows:

Start with $X(0) = 0$, $Y(0) = 0$, $X(1) = 1$, $Y(1) = 2$.

For $n > 1$, choose the least positive integers $Y(n) > X(n)$ such that neither $Y(n)$ nor $X(n)$ appear in $\{Y(k) : 1 \leq k < n\}$ or $\{X(k) : 1 \leq k < n\}$ and such that $Y(n) - X(n)$ does not appear in $\{Y(k) - X(k) : 1 \leq k < n\}$ or $\{Y(k) + X(k) : 1 \leq k < n\}$.

Proving conjectures by guessing the automaton

There is no known foolproof way to take a definition like this and directly turn it into an automaton computing the sequence.

However, in this case, we can "guess" an automaton for it as follows.

First, we decide on an appropriate numeration system. In this case, it is already known that this sequence is related to the Tribonacci numbers (see the OEIS description), so it is reasonable that we should use the Tribonacci numeration system.

Proving conjectures by guessing the automaton

We then represent pairs (n, x) in the Tribonacci numeration system, padding the shorter sequence with leading zeros, if needed.

A pair is *valid* if $x = X(n)$.

Say two strings y and z are equivalent if (yw is valid iff zw is valid), for all w of length $\leq i$, for some fixed i. We used $i = 6$.

Do a breadth-first search on the set of all possible strings, identifying the (finitely many) equivalence classes. One can then construct an automaton out of these equivalence classes.

Proving conjectures by guessing the automaton

When we do this, we find a Tribonacci automaton of 27 states for $X(n)$ and 30 states for $Y(n)$. This is our guess.

Now comes the important part: we use `Walnut` to verify that our guess is correct.

We can do this using mathematical induction. We say that a triple (n, x, y) is *good* if all of the following conditions hold:

1. $y > x$;
2. $x \notin \{X(k) : 1 \leq k < n\}$
3. $y \notin \{X(k) : 1 \leq k < n\}$
4. $x \notin \{Y(k) : 1 \leq k < n\}$
5. $y \notin \{Y(k) : 1 \leq k < n\}$
6. $y - x \notin \{Y(k) - X(k) : 1 \leq k < n\}$
7. $y - x \notin \{Y(k) + X(k) : 1 \leq k < n\}$

Proving conjectures by guessing the automaton

To carry out the induction proof we must show three things:

1. The triple $(n, X(n), Y(n))$ is good for all $n \geq 1$;
2. If (n, x, y) is good then $x \geq X(n)$;
3. If $(n, X(n), y)$ is good then $y \geq Y(n)$.

The latter two conditions ensure that each value of $X(n)$ and $Y(n)$ chosen iteratively is indeed the minimal possible value among good candidates.

Proving conjectures by guessing the automaton

This verification can be carried out by the following Walnut code.

```
def good "?msd_trib y>x &
   (~Ek k<n & $xaut(k,x)) &
   (~Ek k<n & $xaut(k,y)) &
   (~Ek k<n & $yaut(k,x)) &
   (~Ek k<n & $yaut(k,y)) &
   (~Ek,a,b k<n & $xaut(k,a) & $yaut(k,b) & y-x=b-a) &
   (~Ek,a,b k<n & $xaut(k,a) & $yaut(k,b) & y-x=b+a)":
eval check1 "?msd_trib An,x,y (n>=1 & $xaut(n,x) & $yaut(n,y)) =>
   $good(n,x,y)":
eval check2 "?msd_trib An,x,y (n>=1 & $good(n,x,y)) =>
   (Ez $xaut(n,z) & x>=z)":
eval check3 "?msd_trib An,x,y (n>=1 & $xaut(n,x) & $good(n,x,y)) =>
   (Ez $yaut(n,z) & y>=z)":
```

and the last three commands all return TRUE.

Proving conjectures by guessing the automaton

Now that we've verified the automaton, we're ready to prove Cassaigne's conjecture.

Theorem. *For all k the sum $X(k) + Y(k)$ equals either $X(Y(k))$ or $Y(X(k))$.*

```
eval julien1 "?msd_trib An,x,y,xy,yx ($xaut(n,x) &
   $yaut(n,y) & $xaut(y,xy) & $yaut(x,yx)) =>
   (xy=x+y|yx=x+y)":
```

And `Walnut` returns `TRUE`. The conjecture is proved.

Other capabilities of `Walnut`

`Walnut` can also count things. In some cases it can find what is known as a *linear representation* for a function.

A linear representation for a function $f : \mathbb{N} \to \mathbb{N}$ is a triple of the form (v, γ, w), where

- v is a $1 \times r$ matrix,
- $\gamma(a)$ is an $r \times r$ matrix for all a
- w is an $r \times 1$ matrix

and $f(n) = v\gamma(x)w$ for all representations x of n. Here the representations can be base k, Fibonacci, Tribonacci, etc.

Other capabilities of Walnut

Linear representations can often be used to prove theorems about $f(n)$ and its growth rate.

For example, let's evaluate $\rho(n)$, the number of distinct length-n blocks appearing in the Thue-Morse sequence $\mathbf{t}$.

```
def equalblock "At (t<n) => T[i+t]=T[j+t]":
def novelblock "Ak (k<i) => ~$equalblock(i,k,n)":
def countblock n "$novelblock(i,n)":
```

Other capabilities of `Walnut`

`Walnut` outputs a linear representation in a form that Maple can understand:

$$v = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \gamma(0) = \begin{bmatrix} 1&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&2&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&2&0 \end{bmatrix}$$

$$\gamma(1) = \begin{bmatrix} 0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&2&0&0&0&0&0 \\ 0&0&2&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&2&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&2&0&0 \end{bmatrix} \quad w = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

With this representation we can quickly compute $\rho(n)$, the number of distinct length-n blocks in $\mathbf{t}$, very efficiently.

Other capabilities of Walnut

Furthermore, we can compute the exact value of $\rho(2^k)$ as follows:

We have

$$\rho(2^k) = v \cdot \gamma(1) \cdot \gamma(0)^k \cdot w,$$

and the minimal polynomial of $\gamma(0)$ is $X^2(X-1)(X-2)$.

This means that $\rho(2^k) = a + b \cdot 2^k$ for some constants a, b and $k \geq 2$.

We can now use the linear representation to compute $\rho(2^k)$ for $k = 2, 3$ and solve for a, b. We have $\rho(4) = 10$ and $\rho(8) = 22$.

Hence $a = -2$, $b = 3$. So $\rho(2^k) = 3 \cdot 2^k - 2$ for $k \geq 2$.

Common mistakes when using Walnut

- Watch out for edge cases. Sometimes a theorem is true for all $n \geq 1$ but fails for $n = 0$.
- Don't use logical assertions with variables in the wrong order. The order of arguments in a multi-variable assertion is alphabetical order of the variable names used to define it.
- Since the domain of variables is understood to be $\mathbb{N}$, the natural numbers, subexpressions that give negative numbers can cause incorrect results. All subexpressions must be non-negative.

Tips for using Walnut

- There are often different ways to translate the same logical statement into Walnut. Some can take much longer to translate than others.
- There are often multiple characterizations of the same property. Some may be first-order expressible, some may not.
- Sometimes being more general takes much more time and space than being specific.

A final word

`Walnut` is free and downloadable from
`https://cs.uwaterloo.ca/~shallit/walnut.html`.

If you use it to solve a problem, please let me know about it!