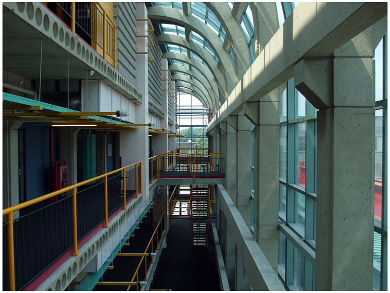
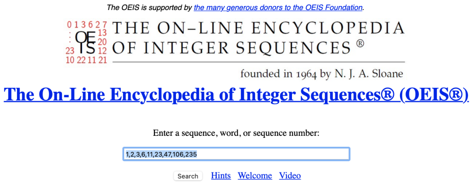
Proving Results About OEIS Sequences with `Walnut`

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 OEIS is supported by the many generous donors to the OEIS Foundation.

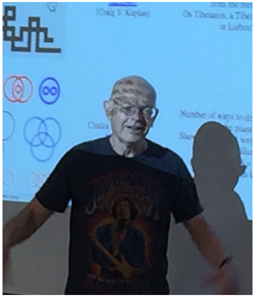
THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES ®

founded in 1964 by N. J. A. Sloane

The On-Line Encyclopedia of Integer Sequences® (OEIS®)

Enter a sequence, word, or sequence number:

1,2,3,6,11,23,47,106,235



Neil Sloane

The *On-Line Encyclopedia of Integer Sequences* (OEIS) is an enormous database of mathematical information, containing over 364,000 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
- talk about its limitations

What is Walnut?

Hamoon Mousavi

- 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 70 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.

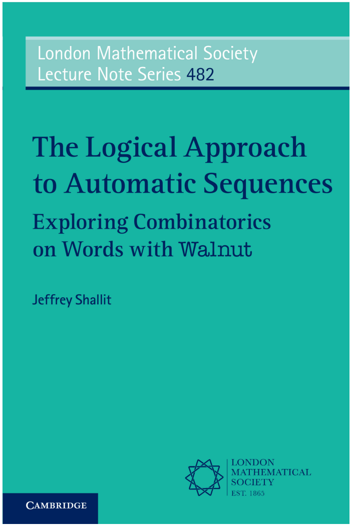
- *But not all sequences!* Just a special class called (generalized) automatic sequences.
    - Examples of sequences in this class include the Thue-Morse sequence, the Rudin-Shapiro sequence, the infinite Fibonacci word, the infinite Tribonacci word, Sturmian words, paperfolding words, overlap-free words etc.
- *But not all theorems!* You have to state the theorem in first-order logic, and you can only use operations such as addition, subtraction, comparison of natural numbers, and indexing 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.


London Mathematical Society
Lecture Note Series 482

The Logical Approach to Automatic Sequences

Exploring Combinatorics on Words with Walnut

Jeffrey Shallit

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:

$$\begin{aligned}\text{odd}(n) &:= \exists k \;\; n = 2k+1 \\ \text{even}(n) &:= \exists k \;\; n = 2k.\end{aligned}$$

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

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

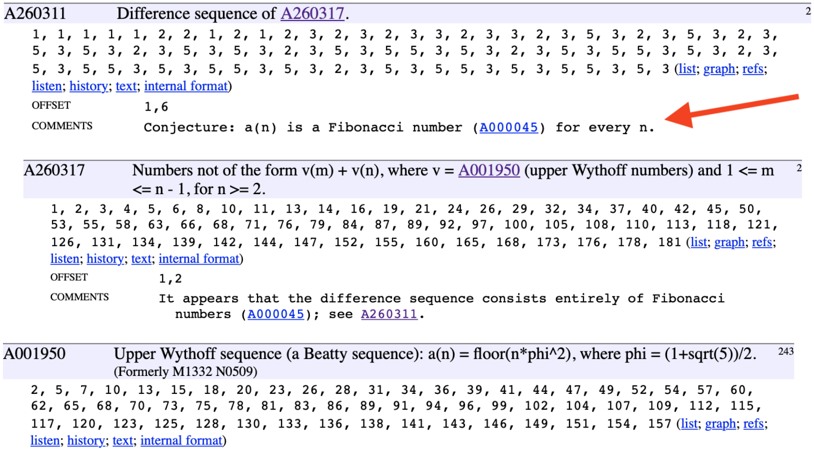
```
[Walnut]$ def odd "Ek n=2*k+1";
[Walnut]$ def even "Ek n=2*k";
[Walnut]$ eval thm "Am,n ($odd(m) & $odd(n)) => $even(m+n)":
(odd(m))&odd(n))):2 states - 3ms
 ((odd(m))&odd(n)))=>even((m+n)))):1 states - 2ms
  (A m , n ((odd(m))&odd(n)))=>even((m+n))))):1 states - 1ms
Total computation time: 33ms.

----
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. 243
(Formerly M1332 N0509)

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 boolean 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 $\mathrm{a}260317(z)$ is true iff z belongs to A260317.

Solving the conjecture

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

In other words, $\mathrm{a260317}(z)$ should be TRUE iff

$$\begin{gathered}\text{there } \textit{do not} \text{ exist } m, n, x, y \\ \text{such that } \texttt{phi2n}(m, x) \text{ and } \texttt{phi2n}(n, y) \\ \text{and } z = x + y \\ \text{and } 1 \leq m \text{ and } m \leq n - 1 \\ \text{and } n \geq 2.\end{gathered}$$

So we just write this as a first-order `Walnut` statement:

```
def a260317 "?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 determine the gaps we say that

there exist t, v such that
$t < v$
and $\mathrm{a260317}(t)$ holds
and $\mathrm{a260317}(v)$ holds
but for all u between t and v, the assertion $\mathrm{a260317}(u)$ does not hold,
and the gap size $g = v - t$.

```
def gap "?msd_fib Et,v t<v & $a260317(t) & $a260317(v) &
    (Au (u>t & u<v) => ~$a260317(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!

How did we get `phi2n`?

We obtained the automaton for `phi2n` using a theorem in a paper of Don Reble in the OEIS! (https://oeis.org/A007895/a007895.pdf)

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.*

What's Fibonacci representation? Here we express a natural number as a sum of distinct Fibonacci numbers, where no two consecutive Fibonacci numbers are used.

This can be represented as a binary string coding which Fibonacci numbers appear in the sum. Example: $20 = F_7 + F_5 + F_3 = 13 + 5 + 2$ is represented by the binary string 101010.

How does it work?

Internally, assertions such as `gap` and `a260317` 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?

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

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).

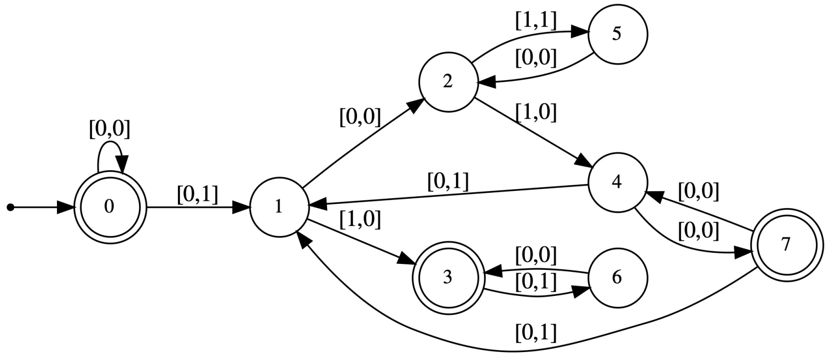
It processes the symbols of the input one-by-one, following the arrow labeled with the symbol.

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.

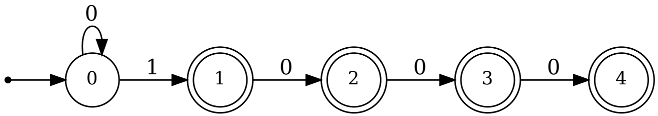


For example, phi2n(10,26) is true. Check with input $[0,1][0,0][1,0][0,1][0,0][1,0][0,0]$. ($[0010010]_F = 10$; $[1001000]_F = 26$)

Going further

So we proved that all members of the gap sequence A260311 are Fibonacci numbers.

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



By examining that automaton, we actually obtain a new result:

Theorem. *The only possible gaps in A260311 are* $1, 2, 3, 5$.

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.

We'll see an example of this in the next slide.

Another example: Shevelev's problem

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

Shevelev's problem was about the Thue-Morse sequence

$$\mathbf{t} = t(0)t(1)t(2)\cdots = 0110100110010110\cdots .$$

This famous sequence is defined as $t(i) =$ the parity of the number of 1-bits in the base-2 representation of i.

Another example: Shevelev's problem

Shevelev observed that for the Thue-Morse sequence there do not exist **two** integers $0 < i < j$ such that

$$t(n) \in \{t(n+i),\ t(n+j)\}$$

for all 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])":
```

and `Walnut` returns `FALSE`.

Another example: Shevelev's problem

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

$$t(n) \in \{t(n+i),\ t(n+j),\ t(n+k)\}$$

for all n.

Shevelev asked for a characterization of 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: Shevelev's problem

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.*

Limitations

- Can't use `Walnut` to prove Lagrange's theorem that all integers are a sum of four squares:

  $$\forall n \ \exists w, x, y, z \ \ n = w^2 + x^2 + y^2 + z^2$$

  because it uses an unsupported operation (squaring).
  - Indeed, as soon as we allow squaring, we get an undecidable logical theory.
- Can't use `Walnut` to prove Goldbach's conjecture:

  $$\forall n \ (n \geq 2) \implies \exists p, q \ 2n = p + q \ \wedge \ \texttt{prime}(p) \ \wedge \ \texttt{prime}(q)$$
  - ...because there is no way to express $\texttt{prime}(p)$ in the logical theory.

Tips for using `Walnut`

- Using the Myhill-Nerode theorem, one can often “guess” an automaton that generates a sequence, and then verify its correctness 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!