THE GUESS-AND-PROVE PARADIGM IN ACTION

MANUEL KAUERS *

Abstract. We give some explicit examples of how computer algebra techniques (or more accurately: computer algebra systems) can be used to solve problems by first guessing a solution and then proving its correctness.

1. Introduction

Modern computer algebra systems offer a lot of useful and nontrivial functionality to their users. Not in all cases is it obvious how to take advantage of these possibilities. In the recent years, we have seen an increasing number of papers about results which rely on an essential use of computer algebra. In such papers the relevant computations are often described only on a conceptual level, with the main goal being to justify that the computations performed really allow to draw the conclusions drawn. Some readers of such papers have difficulties to translate these conceptual descriptions into actual sessions of computer algebra systems, and wonder what exactly they need to type in order to perform the described computations.

In the present paper, we want to address this question by giving some examples. Of course, there is hardly any need for examples when the computer algebra system provides a command that exactly serves the required purpose. For example, commands for factoring polynomials are so easy to use that there is not much more to say about it beyond the examples given in the documentation. Of course the technology behind these commands is highly nontrivial, but this is not our concern here; see, e.g., Chapter 8 of [28] for some recent developments in this direction.

We shall focus here on some applications of computer algebra that involve the combination of several features, or of functions whose purpose may not be immediately obvious from their specification. Obviously, our choice is highly biased and in no way representative. As a common pattern, our examples follow the principle of *guess and prove,* which was already propagated by Polya [32], and which has recently seen some exciting applications in computer algebra, especially in the context of enumerating restricted lattice walks (see [2] and the references given there for lots of examples). The basic idea is to split the work into two phases. In the first phase, the guessing step, we use non-rigorous computations in order to construct a plausible conjecture for the final result or some intermediate auxiliary data. In a second step, the proving step, we use rigorous computations to prove that the conjectures are indeed correct.

Once again, this paper is not meant as a general introduction to computer algebra, nor as an introduction to a certain computer algebra systems. Instead, we hope that the examples given in this paper may serve as inspirations to readers who are already using computer algebra, and who would like to use it more effectively. For a general introduction to computer algebra, readers are referred to the standard textbooks [8, 36]. For introduction to particular computer algebra systems, readers should consult the respective documentation. As we do not want to give a preference to any particular system and therefore include example sessions for Mathematica, Maple, and Sage.

2. Expressions

The usual idea behind automated guessing is to generate lots of necessary conditions on the shape of a hypothetical solution, and then find objects satisfying these conditions. If the conditions are sufficiently overconstrained, it is likely that only solutions of the original problem satisfy them.

A very simple application of this idea is polynomial interpolation. If a given problem has a solution $f(x)$ and if the problem is such that we can compute for every specific choice of x the value $f(x)$, and if we suspect that $f(x)$ is a polynomial, then we can compute $f(1), f(2), \ldots, f(100)$ and interpolate. If the degree of the interpolating polynomial is significantly smaller than 100, this is a strong indication that this polynomial is the answer to the problem.

* Supported by the Austrian FWF grants Y464-N18 and F5004.

Example 1. *Consider the sequence $f(n)$ which is recursively defined by $f(0) = -10, f(1) = -4$, and*

$$f(n) = \frac{450f(n-1)^2 + 450f(n-2)f(n-1) - 225f(n-2)^2 - 9476f(n-2) - 12428f(n-1) - 298444}{675f(n-2) - 1654}$$

for $n \geq 2$. Is this a polynomial sequence? We can guess a candidate by computing the first few terms of the sequence using the recurrence and then interpolating them. In Sage, this can be done as follows.

```
f = [-10, -4]
```

```
for n in range(2, 20) : f.append((450 * f[n - 1]^2 + 450 * f[n - 2] * f[n - 1] - 225 * f[n - 2]^2 - 9476 * f[n - 2] - 12428 * f[n - 1] - 298444)/(675 * f[n - 2] - 1654))
```

```
n = ZZ['n'].gen()
```

```
crt(f, [n - i for i in range(len(f))])
```

$5 * n^3 - 7 * n^2 + 8 * n - 10$

Generically we would have expected a polynomial of degree 20, and with very ugly coefficients. The fact that we got a low degree polynomial with very short coefficients suggests that this polynomial is not a computational artifact but has in fact some significance for the sequence f. In fact, it is equal to f not only for $n = 0, \ldots, 19$ (as it is by construction) but also for all $n \geq 20$. To prove this, we just need to plug the polynomial into the recurrence and simplify.

```
f = _
```

```
-f(n) + (450 * f(n - 1)^2 + 450 * f(n - 2) * f(n - 1) - 225 * f(n - 2)^2 - 9476 * f(n - 2) - 12428 * f(n - 1) - 298444)/(675 * f(n - 2) - 1654)
```

0

In order to guess non-polynomial expressions, we may be able to use linear algebra.

Example 2. *A while ago, we have posed a Monthly problem* [11] *to show that*

$$\sum_{k=0}^{\infty} \frac{F_{3^k} - 2F_{3^k+1}}{F_{3^k} + iF_{2\cdot 3^k}} = i + \tfrac{1}{2}(1 - \sqrt{5}),$$

where F_n is the nth Fibonacci number and $i = \sqrt{-1}$. The problem is easy to solve once we observe that the truncated series admits a closed form. In order to find it, we may first suspect that such a form may have a similar form as the summand expression, say a rational function in $F_{3^k}, F_{3^k+1}, F_{2\cdot 3^k}, F_{2\cdot 3^k+1}$. Writing $f(n) = \sum_{k=0}^{n} \frac{F_{3^k} - 2F_{3^k+1}}{F_{3^k} + iF_{2\cdot 3^k}}$, we seek constants $c_0, \ldots, c_7$ such that

$$f(n) = \frac{c_0F_{3^n} + c_1F_{3^n+1} + c_2F_{2\cdot 3^n} + c_3F_{2\cdot 3^n+1}}{c_4F_{3^n} + c_5F_{3^n+1} + c_6F_{2\cdot 3^n} + c_7F_{2\cdot 3^n+1}}$$

for all $n \in \mathbb{N}$. By clearing the denominator and evaluating the equation for $n = 0, \ldots, 10$, we obtain an overdetermined linear system for the unknown coefficients c_i, the solutions of which give rise to candidate closed forms for the sum.

```
In[1]:= F = Fibonacci;
In[2]:= f[n_Integer] := Sum[(F[3^k] - 2F[3^k + 1])/(F[3^k] + I F[2 * 3^k]), {k, 0, n}]
In[3]:= Solve[Table[(c[4]F[3^n] + c[5]F[3^n + 1] + c[6]F[2 * 3^n] + c[7]F[2 * 3^n + 1])f[n] == (c[0]F[3^n] + c[1]F[3^n + 1] + c[2]F[2 * 3^n] + c[3]F[2 * 3^n + 1]), {n, 0, 10}]]
Out[3]= {{c[2] → I c[0] + I c[1], c[3] → −I c[1], c[4] → (1 + I)c[0] − c[1], c[5] → −c[0] + I c[1], c[6] → (−1 + I)c[0] − (1 − 2I)c[1], c[7] → −I c[0] + (1 − I)c[1]}}
```

We obtain a solution space of dimension two. Specific elements are obtained by setting the free parameters $c[0]$ and $c[1]$ to specific numbers. One possible choice is

```
In[4]:= (c[0]F[3^n] + c[1]F[3^n + 1] + c[2]F[2 * 3^n] + c[3]F[2 * 3^n + 1])/(c[4]F[3^n] + c[5]F[3^n + 1] + c[6]F[2 * 3^n] + c[7]F[2 * 3^n + 1]) /. First[%] /. {c[0] → 1, c[1] → 0}
```

$$\text{Out[4]=} \quad \frac{(1+I)\text{Fibonacci}[3^n] - (1-I)\text{Fibonacci}[2\cdot 3^n] - \text{Fibonacci}[1+3^n] - I\,\text{Fibonacci}[1+2\cdot 3^n]}{\text{Fibonacci}[3^n] + I\,\text{Fibonacci}[2\cdot 3^n]}$$

How can we prove that this guess is correct? If $e(n)$ denotes this expression, it suffices to show that $e(n) - e(n-1)$ is equivalent to the summand expression, and to compare one initial value. The difference of

$e(n)-e(n-1)$ is a rational function in Fibonacci expressions with exponential argument. A convenient way of showing that it is identically zero is via Binet's formula, using in addition the fact that $(-1)^{3^n} = -1$ for all $n \in \mathbb{N}$. To avoid fractional arguments, we compare $e(n+1)-e(n)$ to the shifted summand expression.

```
In[5]:= e[n_] = %;
In[6]:= e[n + 1] - e[n] - ((F[3^n] - 2F[3^n + 1])/(F[3^n] + I F[2 * 3^n]) /. n → n + 1) /.
          Fibonacci[u_. + v_.3^w_.] → 1/Sqrt[5](GoldenRatio^(u+v3^w) - (-1)^(u+v)/GoldenRatio^(u+v3^w)) // Together
Out[6]= 0
In[7]:= e[0] == ((F[3^n] - 2F[3^n + 1])/(F[3^n] + I F[2 * 3^n]) /. n → 0)
Out[7]= True
```

At this point we have guessed-and-proved the formula

$$\sum_{k=0}^{n} \frac{F_{3^k} - 2F_{3^k+1}}{F_{3^k} + iF_{2\cdot 3^k}} = \frac{(1+I)F_{3^n} - (1-I)F_{2\cdot 3^n} - F_{3^n+1} - i\,F_{2\cdot 3^n+1}}{F_{3^n} + i\,F_{2\cdot 3^n}}$$

for $n \in \mathbb{N}$. Now the limit can easily be computed by hand. Also Mathematica is able do it, if we replace 3^n by a fresh variable m.

```
In[8]:= Limit[e[n] /. 3^n → m, m → Infinity]
Out[8]= ((1 + 2 I) - Sqrt[5])/2
In[9]:= Clear[F, f, e]
```

Using the representation as linear combination of exponentials, many formulas about Fibonacci numbers can be proved automatically. Relations among certain quantities satisfying more complicated recurrence equations can also be proven automatically. For a class of sequences defined by nonlinear recurrences, an algorithm was given by the author [10]. With the corresponding Mathematica package [12], all the steps carried out in the example above can be condensed into a single command.

Other tools are needed to find other types of expressions. For example, formulas for determinants tend to involve nested products, and such expressions can be guessed using a package of Krattenthaler [24] for Mathematica, or more recent code by Hebisch and Rubey [9] for FriCAS.

Example 3. *For every $n \in \mathbb{N}$, consider the determinant of the matrix $A = ((a_{i,j}))_{i,j=1}^{n}$ with $a_{i,j} = \frac{1}{i+j}$. It is easy to compute these determinants for the first few values of n. Krattenthaler's package can find a product representation for this data.*

```
In[10]:= << Rate.m
In[11]:= Table[Det[Table[1/(i + j), {i, 1, n}, {j, 1, n}]], {n, 1, 10}]
Out[11]= {1/2, 1/72, 1/43200, 1/423360000, 1/67212633600000, 1/1721536003934208000000,
          1/709706385248124486942720000000,
          1/4702142622508202833251304734720000000000, 1/5001937035648605871126851505665448345600000000000,
          1/8537000898240926708833515201784986712482596782080000000000000}
In[12]:= Rate@@%
Out[12]= {(Product_{i1=1}^{i0-1} (Product_{i2=1}^{-1+i1} (1 + i2)(2 + i2)/(4(3 + 2 i2)^2))/36)/2}
```

We have thus obtained the conjecture

$$\begin{vmatrix} \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n} \end{vmatrix} \stackrel{?}{=} \frac{1}{2} \prod_{k=1}^{n-1} \left(\frac{1}{36} \prod_{i=1}^{k-1} \frac{(i+1)(i+2)}{4(2i+3)^2} \right)$$

We will see later how to prove this conjecture.

3. Equations

Instead of trying to match expressions against data, we can also search for functional equations (e.g., differential or recurrence equations) which are consistent with the given data. There are a lot of tools for doing so, and they are very popular. There are implementations for Maple [33], Mathematica [13], Sage [17], or FriCAS [9]. Let us see an example session in Maple.

Example 4. *Suppose we want to find out whether the solutions of the differential equation*

$$24f(x) - (24x+8)f'(x) + (27x^3 + 21x^2 - 7x - 1)f''(x) = 0$$

are algebraic functions. One way of doing so is by calling the dsolve command of Maple. It will have no trouble finding the desired solutions. Although there are algorithms which for any given linear differential equation with polynomial equations can determine the algebraic solutions, no computer algebra systems currently has a complete implementation of such an algorithm. Even the powerful solver of Maple will overlook algebraic solutions when the differential equation is too large.

When this happens, we can try the guess-and-prove approach. We can first compute series solutions of the differential equation and then use suitable commands of the gfun package to guess an algebraic equation for this solution. To keep the expressions small, let's illustrate this with the differential equation above, even though this equation is small enough that Maple can solve it directly.

with(gfun) :
$deq := 24 * f(x) - (24 * x + 8) * \mathrm{diff}(f(x), x) + (27 * x^3 + 21 * x^2 - 7 * x - 1) * \mathrm{diff}(f(x), x, x)$:
Order := 10 : dsolve($\{deq, f(0) = 1234, D(f)(0) = 5678\}, f(x)$, series);

$$1345 + 2345x + 6760x^2 - 33800x^3 + 196040x^4 - 1250600x^5 + 8497320x^6 - 60333000x^7 + 442408200x^8 - 3324737000x^9 + \mathrm{O}(x^{10})$$

seriestoalgeq(%, $f(x)$);

$$[-36738675x^2 - 21636350x - 4082075 + (13140x + 4380)f(x) - f(x)^2, ogf]$$

The command seriestoalgeq has found the minimal polynomial of an algebraic function which fits to the first few terms of a series solution of the differential equation. In order to prove that this guess is correct, we can use another command of gfun to construct a differential equation for the algebraic function defined by this minimal polynomial. If the guess is right, this differential equation must match the equation we started with. (More precisely, it must be contained as a right factor.)

diffeqtohomdiffeq(algeqtodiffeq(%[1], $f(x)$), $f(x)$);

$$\{-105120f(x) + (35040 + 105120x)\frac{d}{dx}f(x) + (-118260x^3 - 91980x^2 + 30660x + 4380)\frac{d^2}{dx^2}f(x),$$

$$f(0) = \mathrm{RootOf}(\_Z^2 - 4380\_Z + 4082075), D(f)(0) = 5\mathrm{RootOf}(\_Z^2 - 4380\_Z + 4082075) - 4380\}$$

normal(%[1]/deq);

$$-4380$$

This proves that the differential equation satisfied by the guessed algebraic function is a constant multiple of the given differential equation. Therefore, the two differential equations have the same solutions, and in particular the guessed algebraic function is a solution of the original differential equation.

In the example above, the guessing part is hidden in the command seriestoalgeq, which takes a truncated series and returns, a small algebraic equation which fits to the given terms, if there is one. This command is part of the gfun package. The same package also provides the commands which are used in the proving part. These algorithms apply to the class of D-finite functions. A function is D-finite if it satisfies a linear differential equation with polynomial coefficients. As the solution of a differential equation is uniquely determined by the first few initial terms of its series expansion, we can use the differential equation together with the initial values as a finite data structure for representing the function. The commands provided by the gfun package make it possible to perform various operations on this representation. See [34, 33, 35, 18, 15] for the relevant theory.

The theory also applies to sequences satisfying recurrence equations, as well as functions or sequences in several variables. For problems involving several variables, it is more natural to work with linear operators instead of functional equations. These operators live in certain non-commutative algebras, and the set of all operators that map a particular function to zero forms a left ideal in that algebra, called the annihilating ideal. A multivariate D-finite function is uniquely determined by its annihilating ideal and finitely many initial values. Computations with such ideals of operators can be performed with the Maple package mgfun [6], the Mathematica package HolonomicFunctions.m [19, 20], or with the Sage package ore_algebra [17].

An important technique in the multivariate case is creative telescoping [37, 31, 18, 21, 5, 16], which can be used for finding recurrence equations satisfied by definite sums and integrals. As an illustration of this technique, we will show how to prove the determinant evaluation conjectured in Example 3. The proof relies on an approach for proving determinant evaluations proposed by Zeilberger [38]. This technique was used in our proof of the qTSPP conjecture [22] and for other examples [23]. The observation is that

in order to prove $\det((a_{i,j}))_{i,j=1}^n = b_n$, it suffices to exhibit a certificate sequence $c_{n,k}$ with the following three properties:

- $c_{n,n} = 1$ for all $n \geq 1$
- $\sum_{k=1}^n a_{i,k} c_{n,k} = 0$ for all $1 \leq i < n$
- $\sum_{k=1}^n a_{n,k} c_{n,k} = b_n / b_{n-1}$ for all $n \geq 1$ (taking $b_0 = 1$).

These properties together with the initial value $a_{1,1} = b_1$ imply the determinant identity. (See [38, 22, 23] for a justification.)

Example 5. *First we have to guess a certificate $c_{n,k}$. To generate data, we can compute some of these terms. Observe that $(c_{n,1}, \ldots, c_{n,n})$ is the last row of $\frac{\det(A_n)}{\det(A_{n-1})} A_n^{-1}$.*

```
from ore_algebra import *
```

```
A = lambda n : matrix(QQ, [[1/(i+j)for j in range(1, n+1)] for i in range(1, n+1)])
```

```
data = [list(A(n).det()/A(n-1).det() * A(n).inverse()[n-1]) + [0]*(20-n) for n in range(1, 21)]
```

```
n, k = ZZ['n','k'].gens()
```

```
Alg.<Sn, Sk> = OreAlgebra(ZZ[n,k])
```

```
guess(data, Alg, order=1).groebner_basis()
```

$[(k^2 + 3*k + 2)*Sk + n^2 - k^2 + 2*n - 2*k,$
$(4*n^2 - 4*n*k + 10*n - 6*k + 6)*Sn + n^2 + n*k + 4*n + 2*k + 4]$

```
annC = Alg.ideal([g.map_coefficients(lambda p : p(n-1, k-1)) for g in _])
```

```
annC
```

Left Ideal $((k^2 + k)*Sk + n^2 - k^2, (4*n^2 - 4*n*k + 6*n - 2*k + 2)*Sn + n^2 + n*k + n + k)$ of Multivariate Ore algebra in Sn, Sk over Fraction Field of Multivariate Polynomial Ring in n, k over Integer Ring

We are lucky that a small system of operators has been found. Its solution is uniquely determined up to a constant multiplicative factor. If we want, we can express the solution in terms of binomial expressions. This requires a little bit of fiddling, and is not really necessary. In any case, one possible expression is

$$c_{n,k} = 2(-1)^{n+k} \binom{n-1}{k-1} \binom{n+k-1}{k} / \binom{2n}{n}.$$

The multiplicative factor 2 is chosen so as to match the data computed before:

```
c = lambda n, k : 2*(-1)^(n+k)*binomial(n-1, k-1)*binomial(n+k-1, k)/binomial(2*n, n)
```

```
[[c(n, k) for k in range(1, 21)] for n in range(1, 21)] == data
```

True

To complete the proof of the determinant identity, it remains to prove that the guessed sequence $c_{n,k}$ has the three required properties. While it is fairly easy to see that $c_{n,n} = 1$ is true for $n \geq 1$, the other properties are less obvious. In order to prove them, we use creative telescoping. The mathematical reasoning behind these steps are not self-evident, the reader not familiar with creative telescoping may want to consult one of the many tutorials on this work (e.g., [31, 18]*)*

$i,n,k = ZZ['i','n','k'].\text{gens}()$

$Alg2.\langle Si, Sn, Sk\rangle = \text{OreAlgebra}(ZZ[i,n,k])$

$annC = Alg2.\text{ideal}(\text{list}(annC.\text{gens}()) + [Si-1])$

$annAik = Alg2.\text{ideal}([(1+i+k)*Sk-(i+k),(1+i+k)*Si-(i+k),Sn-1])$

$annAik.\text{symmetric\_product}(annC).\text{ct}(Sk-1)[0]$

$[(-4*i*n-4*n^2-2*i-6*n-2)*Sn+i*n-n^2+i-n,(i^2-n^2+2*i+1)*Si-i^2-i]$

$annAnk = Alg2.\text{ideal}([(1+n+k)*Sk-(n+k),(1+n+k)*Sn-(n+k),Si-1])$

$annAnk.\text{symmetric\_product}(annC).\text{ct}(Sk-1)[0]$

$[(-16*n^2-16*n-4)*Sn+n^2+n,Si-1]$

The first of these two outputs says that the sum $f(n,i) := \sum_{k=1}^{n} a_{i,k}c_{k,n}$ *satisfies the recurrence equations*

$$(1+i-n)(1+i+n)f(n,i+1) = i(i+1)f(n,i), \quad 2(1+i+n)(1+2n)f(n+1,i) = (i-n)(1+n)f(n,i),$$

from which it can be shown that $f(n,i) = 0$ *for* $1 \le i < n$.
The second output says that the sum $f(n) := \sum_{k=1}^{n} a_{n,k}c_{k,n}$ *satisfies the recurrence*

$$4(2n+1)^2 f(n+1) = n(n+1)f(n).$$

As this recurrence is also satisfied by b_n/b_{n-1}*, we have completed the proof.*

4. Numbers

Besides expressions and functional equations for sequences or power series, we can also guess information about numbers. The two main techniques for doing so are the general LLL-algorithm [25, 28] and the more specialized PSLQ-algorithm [7].

Example 6. *The equation* $\arctan(x) = (x^2+1)\arctan(2-x)$ *has a unique real solution. No algorithm is known for solving such transcendental equations in terms of closed form numbers, but we can still try guess-and-prove.*

$Digits := 50; alpha := \text{fsolve}(\arctan(x) - (x^2+1)*\arctan(2-x));$

$$Digits := 50$$
$$alpha := 1.73205080756887729352744634150587236694280525381 04$$

with(IntegerRelations);

$$[\text{LLL}, \text{LinearDependency}, \text{PSLQ}]$$

$\text{PSLQ}([1, alpha, alpha^2, alpha^3]);$

$$[-3, 0, 1, 0]$$

This output suggests that $-3+0\alpha+\alpha^2+0\alpha^3 = 0$*, i.e.,* $\alpha = \sqrt{3}$*. To prove that this is indeed the right answer, it suffices to plug it into the equation and rely on Maple's simplifier.*

$\text{simplify}(\text{subs}(x = \text{sqrt}(3), \arctan(x) - (x^2+1)*\arctan(2-x)));$

$$0$$

One of the most spectacular results obtained along these lines is the BBP formula for π, which was accidentally discovered during a search for integer relations among infinite series. This formula was totally unexpected, but once it was discovered, its proof was not very difficult: it is equivalent to a linear combination of certain rational integrals which a computer algebra system can easily evaluate. See [1] for further details on this story.

Guessing is often easier than proving, especially for problems about numbers. We frequently encounter such situations when we want to determine the asymptotic behaviour of certain combinatorial sequences. We know that these sequences behave asymptotically like $c\rho^n n^\alpha$ for certain constants c, ρ, α, but while ρ and α are reasonably easy to determine, but we do not know how to compute c (unless the methods of Pemantle and Wilson apply [29, 30]). Still, we can often guess the right value of c. We can compute ρ and α from a recurrence for the sequence (which itself may be guessed), then use them to compute a numerical approximation of c to high precision, and finally recover the exact value from the approximation. We conclude by giving one example computation of this sort. See [4, 3, 26, 27] for more.

Example 7. *The sequence $f(n)$ is defined by $f(0) = 1$, $f(1) = 5$, and*

$$36(n+1)(2n+1)(2n+3)f(n) - 10(n+2)(2n+3)(2n+5)f(n+1) + (n+2)(n+3)(n+4)f(n+2) = 0$$

for $n \geq 0$. We want to know the asymptotic behaviour of this sequence. Using a package written by the author [14]*, we can compute formal asymptotic solutions of the recurrence.*

```
In[13]:= << Asymptotics.m
         Asymptotics Package by Manuel Kauers  RISC Linz  V 0.11 (2012-07-19)
In[14]:= rec = 36*(1+n)*(1+2*n)*(3+2*n)*f[n] - 10*(2+n)*(3+2*n)*(5+2*n)*f[1+
         n] + (2+n)*(3+n)*(4+n)*f[2+n];
In[15]:= Asymptotics[rec, f[n]]
Out[15]= {4^n/n^3, 36^n/n^3}
```

This output means that for $n \to \infty$ our sequence $f(n)$ should agree with a certain linear combination of $4^n n^{-3}$ and $36^n n^{-3}$. Since the latter dominates the former, we expect $f(n) \sim c36^n n^{-3}$ for some constant c. We can determine the constant numerically by evaluating the quotient $f(n)/(36^n n^{-3})$ for large indices n.

```
In[16]:= f[n_Integer] := Which[n == 0, 1, n == 1, 5, True, f[n] = (-2*(18*(-1+n)*(-3+2*n)*
         (-1+2*n)*f[-2+n] - 5*n*(-1+2*n)*(1+2*n)*f[-1+n]))/(n*(1+n)*(2+n))]
In[17]:= u[n_Integer] := f[n]/(36^n n^-3)
In[18]:= N[u[1000], 50]
Out[18]= 1.5134596447757720515220774543079292467924679843816
In[19]:= N[u[2000], 50]
Out[19]= 1.5163678166038235873110260313758186940468202585729
In[20]:= N[u[3000], 50]
Out[20]= 1.5173389381113803705216478071872594483911360060059
```

Okay, this seems to converge. But the convergence is pretty slow. Despite using several thousand terms, we hardly get two decimal digits of accuracy. This won't be enough to recover the exact value. We can considerably improve the performance by taking the first few terms of the asymptotic expansions into account. These can also be computed with the Asymptotics package.

```
In[21]:= Asymptotics[rec, f[n], Order → 2]
Out[21]= {4^n(1 + 11729/(2048n^2) - 93/(32n))/n^3, 36^n(1 + 21089/(2048n^2) - 123/(32n))/n^3}
In[22]:= expansion[n_] = Last[Asymptotics[rec, f[n], Order → 30]];
In[23]:= u[n_Integer] := f[n]/expansion[n]
In[24]:= N[u[1000], 50]
Out[24]= 1.5192837835151164923743986348238747332464688780494
In[25]:= N[u[2000], 50]
Out[25]= 1.5192837835151164923743986348238747332464688780494
In[26]:= N[u[3000], 50]
Out[26]= 1.5192837835151164923743986348238747332464688780494
```

This is more useful; it looks like we can trust at least the first 50 digits. To recover the exact value from the approximation, we can use Mathematica's builtin LLL engine. The following commands produce the coefficient list of a conjectured minimal polynomial of degree at most three.

```
In[27]:= c = %;
```

```
In[28]:= Most[First[LatticeReduce[
              Transpose[Append[IdentityMatrix[4], Floor[10^50 * c^Range[0, 3]]]]]]]
Out[28]= {-466510788517, 1274558997045, -396763651656, -158001503235}
```

That does not look convincing. Probably the limit is not a cubic algebraic number. We could next try to see if the limit can be written as a product of rational powers of small primes.

```
In[29]:= Most[First[LatticeReduce[
              Transpose[Append[IdentityMatrix[5], Floor[10^50 * Log[{c, 2, 3, 5, 7}]]]]]]]
Out[29]= {1913619642, 5397440880, 2649112157, -1504537155, -2585144737}
```

These can hardly be the right exponents. Maybe we have to allow some other constants to appear in the expression, such as e *or* π*.*

```
In[30]:= Most[First[LatticeReduce[
              Transpose[Append[IdentityMatrix[7], Floor[10^50 * Log[{c, 2, 3, 5, 7, E, Pi}]]]]]]]
Out[30]= {2, 5, -6, 0, 0, 0, 2}
```

This looks much better! It suggests that $c^2 2^5 3^{-6} \pi^2 \approx 1$*, so* $c \approx \frac{27}{\sqrt{32\pi}}$*. By construction, this number matches the numerical approximation to at least 50 decimal digits. It continues to match if we increase the precision, which adds further evidence that this is the correct number.*

```
In[31]:= N[u[3000] - 27/Sqrt[32]/Pi, 50]
Out[31]= -1.5916384284863134212136886738764800892681026709034 10^-88
In[32]:= N[u[5000] - 27/Sqrt[32]/Pi, 50]
Out[32]= -2.1061302299467352839632639627253057878450758432551 10^-95
In[33]:= N[u[7000] - 27/Sqrt[32]/Pi, 50]
Out[33]= -6.2096552325227282669338997935809208070195638383563 10^-100
```

We have no doubt that indeed $f(n) \sim \frac{27}{\sqrt{32\pi}} 36^n n^{-3}$ *(*$n \to \infty$*), but we do not know of any (semi-)automatic method that could help us prove that the guessed multiplicative constant is correct.*

References

[1] David H. Bailey, Peter B. Borwein, and Simon Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66:903–913, 1997.

[2] Alin Bostan. Calcul formel pour la combinatoire des marches. Habilitation a Diriger des Recherches, Universite Paris 13, 2017.

[3] Alin Bostan, Frédéric Chyzak, Mark van Hoeij, Manuel Kauers, and Lucien Pech. Hypergeometric expressions for generating functions of walks with small steps in the quarter plane. *European Journal of Combinatorics*, 61:242–275, 2017.

[4] Alin Bostan and Manuel Kauers. Automatic classification of restricted lattice walks. In *Proceedings of FPSAC'09*, pages 201–215, 2009.

[5] Frédéric Chyzak. *The ABC of Creative Telescoping – Algorithms, Bounds, Complexity*. Habilitation á diriger des recherches. University Paris-Sud 11, 2014.

[6] Frédéric Chyzak. The mgfun package. ttps://specfun.inria.fr/chyzak/mgfun.html, 2016.

[7] Helaman R. P. Ferguson and David H. Bailey. A polynomial time, numerically stable integer relation algorithm. Technical Report RNR-91-032, RNR, 1992.

[8] Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms for Computer Algebra*. Kluwer, 1992.

[9] Waldemar Hebisch and Martin Rubey. Extended Rate, more GFUN. *Journal of Symbolic Computation*, 46(8):889–903, 2011.

[10] Manuel Kauers. *Algorithms for Nonlinear Higher Order Difference Equations*. PhD thesis, Johannes Kepler Universität Linz, 2005.

[11] Manuel Kauers. Problem 11258. *The American Mathematical Monthly*, 113(10):939, December 2006.

[12] Manuel Kauers. SumCracker: A package for manipulating symbolic sums and related objects. *Journal of Symbolic Computation*, 41(9):1039–1057, 2006.

[13] Manuel Kauers. Guessing handbook. Technical Report 09-07, RISC-Linz, 2009.

[14] Manuel Kauers. A Mathematica package for computing asymptotic expansions of solutions of p-finite recurrence equations. Technical Report 11-04, RISC-Linz, 2011.

[15] Manuel Kauers. The holonomic toolkit. In *Computer Algebra in Quantum Field Theory: Integration, Summation and Special Functions*, Texts and Monographs in Symbolic Computation, pages 119–144. Springer, 2013.

[16] Manuel Kauers. Computer algebra. In *Handbook of Enumerative Combinatorics*, pages 975–1046. Taylor and Francis, 2015.

[17] Manuel Kauers, Maximilian Jaroschek, and Fredrik Johansson. Ore polynomials in Sage. In *Computer Algebra and Polynomials*, LNCS 8942, pages 105–125. Springer, 2014.

[18] Manuel Kauers and Peter Paule. *The Concrete Tetrahedron*. Springer, 2011.

[19] Christoph Koutschan. *Advanced Applications of the Holonomic Systems Approach*. PhD thesis, Johannes Kepler University, 2009.

[20] Christoph Koutschan. HolonomicFunctions (User's Guide). Technical Report 10-01, RISC Report Series, University of Linz, Austria, January 2010.
[21] Christoph Koutschan. Creative telescoping for holonomic functions. In *Computer Algebra in Quantum Field Theory: Integration, Summation and Special Functions*, Texts and Monographs in Symbolic Computation, pages 171–194. Springer, 2013.
[22] Christoph Koutschan, Manuel Kauers, and Doron Zeilberger. Proof of George Andrews' and David Robbins' q-TSPP-conjecture. *Proceedings of the National Academy of Sciences*, 108(6):2196–2199, 2011.
[23] Christoph Koutschan and Thotsaporn Thanatipanonda. Advanced computer algebra for determinants. *Annals of Combinatorics*, 17(3):509–523, 2013.
[24] Christian Krattenthaler. RATE: A Mathematica guessing machine. Available at http://mat.univie.ac.at/~kratt/rate/rate.html, 1997.
[25] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Annals of Mathematics*, 126:515–534, 1982.
[26] Stephen Melczer and Marni Mishna. Asymptotic lattice path enumeration using diagonals. *Algorithmica*, 74(4):782–811, 2016.
[27] Stephen Melczer and Marc Wilson. Asymptotics of lattice walks via analytic combinatorics in several variables. In *Proceedings of FPSAC 2016*, pages 863–874, 2016.
[28] Phong Q. Nguyen and Brigitte Vallée. *The LLL Algorithm*. Springer, 2010.
[29] Robin Pemantle and Mark C. Wilson. Twenty combinatorial examples of asymptotics derived from multivariate generating functions. *SIAM Review*, 50(2):199–272, 2008.
[30] Robin Pemantle and Mark C. Wilson. *Analytic Combinatorics in Several Variables*. Cambridge, 2013.
[31] Marko Petkovšek, Herbert Wilf, and Doron Zeilberger. $A = B$. AK Peters, Ltd., 1997.
[32] George Polya. Guessing and proving. *The Two-Year College Mathematics Journal*, 9(1):21–27, 1978.
[33] Bruno Salvy and Paul Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.
[34] Richard P. Stanley. Differentiably finite power series. *European Journal of Combinatorics*, 1:175–188, 1980.
[35] Richard P. Stanley. *Enumerative Combinatorics, Volume 2*. Cambridge Studies in Advanced Mathematics 62. Cambridge University Press, 1999.
[36] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
[37] Doron Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11:195–204, 1991.
[38] Doron Zeilberger. The holonomic ansatz II: Automatic discovery(!) and proof(!!) of holonomic determinant evaluations. *Annals of Combinatorics*, 11(2):241–247, 2007.

Manuel Kauers, Institute for Algebra, J. Kepler University Linz, Austria
E-mail address: manuel.kauers@jku.at