SumCracker: A Package for Manipulating Symbolic Sums and Related Objects

Manuel Kauers [1]

Research Institute for Symbolic Computation
Johannes Kepler Universität
Altenberger Straße 69
4232 Linz, Austria, Europe

Abstract

We describe a new software package, named SumCracker, for proving and finding identities involving symbolic sums and related objects. SumCracker is applicable to a wide range of expressions for many of which there has not been any software available up to now. The purpose of this paper is to illustrate how to solve problems using that package.

Key words: Symbolic Summation, Combinatorial Sequences, Software

1 Introduction

In this paper, we shall introduce a new Mathematica package for symbolic summation. Several packages for this purpose have already been presented in the past. Most prominently, several implementations of the classical hypergeometric and q-hypergeometric summation algorithms [26] are available [24,28,1]. Also for more sophisticated summation problems, there are some software packages available, for instance Schneider's [29] implementation of Karr's algorithm [16,17] in Mathematica. There are some more specialized software packages, too, for instance for identities of Rogers-Ramanujan type [32].

The philosophy of our package is somewhat into the other direction. Rather than a package providing powerful algorithms, restricted to a small domain, SumCracker contains implementations of more general algorithms, which apply to a class of sequences that is very broad. For summation problems to

[1] Partially supported by the Austrian Science Foundation (FWF) grants F1305 and P16613-N12.

which they are applicable, all the implementations mentioned above are superior to ours with respect to strength and efficiency. The advantage of the SumCracker package lies in its ability to treat also very peculiar summation problems, which are out of the scope of all other summation packages that have appeared up to now. Therefore, our package should be activated [only] when the problem at hand does not fit into any of them.

SumCracker can simplify symbolic sums, but not only that. In fact it can simplify any expression that it understands (Section 5.2). It supports conversion operations ("express this in terms of this", Section 5.3), and it is able to find solutions of certain nonlinear difference equations (Section 5.4). All these features rely on a general procedure for discovering algebraic dependencies of given sequences. This procedure is at the heart of the SumCracker package (Section 6). In addition, the package contains a tool for proving identities and inequalities about sequences (Section 4).

The goal of this article is to describe *what* SumCracker can do and to explain how to get it do something, but we do not comment on *how* it obtains its results. The paper is intended as a guide for potential users of the package. The underlying algorithms are described elsewhere [18,19,13,21,20].

The algorithms implemented in the package operate on a class of univariate sequences $\mathbb{N} \to k$, which we call *admissible.* A sequence is admissible if it can be viewed as a solution of a certain type of systems of difference equations, which we call *admissible systems.* The commands provided by the package allow to input admissible sequences by means of a defining admissible system, but the construction of an admissible system is often a cumbersome and errorprone task. Therefore, some effort was put into routines that automatically transform a description of a sequence in terms of a natural expression into a corresponding admissible system. This routines apply to a lot of expressions, and these expressions we also call *admissible.* Such admissible expressions include expressions for special sequences such as Fibonacci[n] or JacobiP[n, a, b, x], and new admissible expressions can be obtained from atomic ones by arithmetic operations, by applying product, summation, or continued fraction operators, and by applying affine transformations to the argument. A precise description of the admissible expressions is given in Section 3.

SumCracker was implemented in Mathematica. It is available free of charge for any non-commercial user and can be obtained from http://www.risc.uni-linz.ac.at/research/combinat/software/ or upon request from the author. If the package file resides at a location where Mathematica finds it, the package can be loaded as follows.

In[1]:= **<< SumCracker.m**

SumCracker Package by Manuel Kauers – © RISC Linz – V 0.2 2005-12-14

Example input and output is typeset as above. The syntax used in the input lines should be precise enough that the examples can be reproduced. Only minor simplifications (such as writing a^b instead of $a\hat{\ }b$) have been employed in input lines to improve readability. With respect to the output, we have decided not to stick to Mathematica's syntax too closely, but to use standard mathematical notation. It should also be noticed that the precise form of the output might be different in future versions of the package.

Unless the runtime of a particular command line is explicitly mentioned, the results are obtained in less than two seconds. Timings are taken with respect to Mathematica 5.2 on a Debian Linux machine with a 2.5 GHz CPU and 2 GB of memory. An asterix at a timing indicates that internal Gröbner basis computations were not carried out by Mathematica's built-in command, but by the special purpose software Singular [15].

2 Motivating Examples

The most simple sequence which is not hypergeometric is probably the sequences of Fibonacci numbers F_n, defined via

$$F_{n+2} = F_n + F_{n+1} \quad (n \geq 0), \qquad F_0 = 0, F_1 = 1.$$

This sequence arises in numerous combinatorial contexts, and there are a lot of identities for this sequence. A nontrivial identity involving Fibonacci numbers concerns the summation problem $\sum_{k=0}^{n} 1/F_{2^k}$ [14, Ex. 6.61].

With our package, we can easily find a closed form for this sum for $n \geq 1$, as follows.

In[2]:= **Crack[SUM[1/Fibonacci[2^k], {k, 0, n}], From → 1]**

Out[2]=
$$\frac{4F_{2^n} - F_{2^n+1}}{F_{2^n}}$$

As a direct consequence, we obtain Mellin's series [23]

$$\sum_{k=0}^{\infty} \frac{1}{F_{2^k}} = \tfrac{1}{2}(7 - \sqrt{5}),$$

because $\lim_{n\to\infty} F_{n+1}/F_n = \phi = \frac{1}{2}(1 + \sqrt{5})$. Let us postpone the detailed explanation of the Crack command to Section 5 and instead investigate now variations of this summation problem.

We requested above that the closed form be valid for $n \geq 1$, and the closed form we obtained is indeed violated for $n = 0$. But there is also a closed form which is valid from $n = 0$ on:

In[3]:= **Crack[SUM[1/Fibonacci[2^k], {k, 0, n}], From → 0]**

Out[3]=
$$\frac{3F_{2^n}F_{2^n+1} - 1 - F_{2^n}^2}{F_{2^n}F_{2^n+1}}$$

Both this and the former identity can be generalized to Fibonacci polynomials $F_n(z)$, defined via

$$F_{n+2}(z) = zF_{n+1}(z) + F_n(z) \quad (n \geq 0), \qquad F_0(z) = 0, F_1(z) = 1.$$

Note that $F_n = F_n(1)$.

In[4]:= **Crack[SUM[1/Fibonacci[2^k, z], {k, 0, n}], From → 1]**

Out[4]=
$$\frac{2}{z} + 1 + z - \frac{F_{2^n+1}(z)}{F_{2^n}(z)}$$

In[5]:= **Crack[SUM[1/Fibonacci[2^k, z], {k, 0, n}], From → 0]**

Out[5]=
$$1 + \frac{2}{z} - \frac{1}{F_{2^n}(z)F_{2^n+1}(z)} - \frac{F_{2^n}(z)}{F_{2^n+1}(z)} \tag{24.61s}$$

Even further generalization is possible. Already Lucas [22] pointed out the general identity

$$\sum_{k=0}^{n} \frac{q^{a2^k}}{u(a2^{k+1})} = q\left(\frac{u(a2^{n+1}-1)}{u(a2^{n+1})} - \frac{u(n-1)}{u(n)}\right) \qquad (a, n \geq 1), \tag{1}$$

which holds for every sequence $u(n)$ satisfying

$$u(n+2) = pu(n+1) - qu(n), \qquad u(0) = 0, u(1) = 1.$$

SumCracker is not able to find this identity in full generality—it returns the sum unevaluated.

In[6]:= **Crack[SUM[$q^{a \cdot 2^k}$/u[$a \cdot 2^{k+1}$], {k, 0, n}], From → 1,**
Where → {u[n+2] == p·u[n+1]−q·u[n], u[0] == 0, u[1] == 1}]

Out[6]=
$$\sum_{k=0}^{n} \frac{q^{a2^k}}{u(a2^{k+1})}$$

However, it does find a closed form representation of this sum for each specific value of a.

In[7]:= **Crack[SUM[q^{2^k}/u[2^{k+1}], {k, 0, n}], From → 1,**
Where → {u[n+2] == p·u[n+1]−q·u[n], u[0] == 0, u[1] == 1}]

Out[7]=
$$p - \frac{u(2^n+1)}{u(2^n)}$$

In[8]:= **Crack[SUM[$q^{3 \cdot 2^k}$/u[$3 \cdot 2^{k+1}$], {k, 0, n}], From → 1,**
Where → {u[n+2] == p·u[n+1]−q·u[n], u[0] == 0, u[1] == 1}]

Out[8]=
$$\frac{p(p^2-2q)}{p^2-q} - \frac{u(3 \cdot 2^{n+1}+1)}{u(3 \cdot 2^{n+1})}$$

In[9]:= **Crack[SUM[$q^{4\cdot 2^k}$/u[4 · 2^{k+1}], {k, 0, n}], From → 1, Where → {u[n+2] == p·u[n+1]−q·u[n], u[0] == 0, u[1] == 1}]**

Out[9]=
$$\frac{p^4-3p^2q+q^2}{p(p^2-2q)}-\frac{u(4\cdot 2^n+1)}{u(4\cdot 2^n)} \qquad (2.57s)$$

In[10]:= **Crack[SUM[$q^{5\cdot 2^k}$/u[5 · 2^{k+1}], {k, 0, n}], From → 1, Where → {u[n+2] == p·u[n+1]−q·u[n], u[0] == 0, u[1] == 1}]**

Out[10]=
$$\frac{p(p^4-4p^2q+3q^2)}{p^4-3p^2q+q^2}-\frac{u(5\cdot 2^n+1)}{u(5\cdot 2^n)} \qquad (3.31s)$$

When we became aware of the above identities involving F_{2^k}, we were wondering whether there are similar summation identities which are not related to formula (1). Using our package, we have found the identities

$$\sum_{k=0}^{n}\frac{F_{3^k}-2F_{3^k+1}}{F_{3^k}+\mathrm{i}\,F_{2\cdot 3^k}}=\frac{(2+\mathrm{i})F_{3^n}-(1+\mathrm{i})F_{3^n+1}-\mathrm{i}F_{2\cdot 3^n}-F_{2\cdot 3^n+1}}{F_{3^n}-F_{3^n+1}+\mathrm{i}F_{2\cdot 3^n+1}}$$
$$\sum_{k=0}^{n}\frac{P_{3^k}-P_{3^k+1}}{P_{3^k}+P_{2\cdot 3^k+1}}=\frac{2P_{2\cdot 3^n+1}+P_{3^n+1}-P_{2\cdot 3^n}}{2(P_{3^n}+P_{2\cdot 3^n+1})}$$
$$\sum_{k=0}^{n}\frac{\psi F_{F_{k+1}}-2\mathrm{i}F_{F_k}}{\mathrm{i}\sqrt{3}\psi+6\mathrm{i}(-1)^{F_k}+\frac{3}{2}\mathrm{i}\psi^2(-1)^{F_{k+1}}}=\frac{\psi F_{F_{n+1}}}{i\sqrt{3}\psi+6\mathrm{i}(-1)^{F_n}+\frac{3}{2}\mathrm{i}\psi^2(-1)^{F_{n+1}}}$$

where $\mathrm{i}=\sqrt{-1}$, $\psi=-\mathrm{i}+\sqrt{3}$ and P_n denotes the nth Pell number, defined via

$$P_{n+2}=2P_{n+1}+P_n, \qquad P_0=0, P_1=1.$$

We believe that these identities have not been published before. Quite in contrast to the case of hypergeometric sums, sum identities involving recurrent sequences of exponential arguments turn out to be extremely rare. In fact, we did not find the above identities by trial and error applications of the Crack command, but by computing algebraic dependencies of the quantities in question and then doing an exhaustive search for telescoping rational functions in the ideal of algebraic dependencies. A similar technique in a simpler situation is described in Section 6.2 below. Besides the mentioned identities we have only found a few more nontrivial ones, but they had quite an unpeasent appearance.

In this introductory section, we have only made use of the Crack command for breaking an expression into an equivalent but “simpler” expression. SumCracker provides in addition commands for proving identities and inequalities and for discovering algebraic dependencies. These commands, along with examples, are introduced in the subsequent sections.

3 Admissible Sequences and Admissible Expressions

Before we turn to a more detailed description of the SumCracker facilities, let us clarify the domain of sequences which SumCracker can handle. The algorithms implemented in the package operate on a certain class of sequences which we call *admissible.* In order to be admissible, a sequence must be a solution of a system of difference equations of a certain type, the *admissible systems.* A precise definition is given below. In order to refer to a certain admissible sequence, the user can directly specify the defining admissible system, but often this is not necessary. SumCracker has got routines that are able to transform a lot of standard expressions into suitable defining admissible systems. The user can therefore input many admissible sequences by expressions in a natural style. An expression which SumCracker is able to recognize as an admissible sequence is called an *admissible expression.* We now state precise definitions for sequences and expressions to be admissible.

Definition 1 *Let* $S = \{\mathit{diffeq}_1, \ldots, \mathit{diffeq}_m\}$ *be a system of difference equations with the function symbols* $f_1, \ldots, f_m$*, where each* diffeq_i *has the form*

$$\begin{aligned} f_i(n+r_i) = \operatorname{rat}_i\Bigl(& f_1(n), \quad f_1(n+1), \quad \ldots, f_1(n+r_i-1), \quad f_1(n+r_i), \\ & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ & f_{i-1}(n), f_{i-1}(n+1), \ldots, f_{i-1}(n+r_i-1), f_{i-1}(n+r_i), \\ & f_i(n), \quad f_i(n+1), \quad \ldots, f_i(n+r_i-1), \\ & \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ & f_m(n), \quad f_m(n+1), \quad \ldots, f_m(n+r_i-1)\Bigr) \end{aligned}$$

with some explicit rational function rat_i*. Then the system* S *is called an* admissible system *(for* $f_1, \ldots, f_m$*).*

A sequence $f\colon \mathbb{N} \to k$ *is called admissible if there exists an admissible system* S *with solutions* $f_1, \ldots, f_m\colon \mathbb{N} \to k$ *such that* $f = f_i$ *for some* $i = 1, \ldots, m$*.*

SumCracker internally represents admissible sequences by using defining admissible systems and a suitable number of initial values. Note that this data uniquely defines the admissible sequence, because the difference equations in an admissible system allow to determine the values $f_i(n)$ if the values $f_i(j)$ ($j < n$) are known. (A problem arises only if the iterated application of a recurrence leads to a division by zero, in which case the sequences are not well defined. We assume that this does not happen in admissible systems which the user specifies.)

The class of admissible sequences is closed under a number of important operations, and for many operations it is easy to get automatically from an admissible system of the operands to an admissible system of the sequence resulting from the operation. Roughly speaking, an admissible expression is an expression that can be obtained from some standard expressions by means of such operations.

Definition 2 *An expression* $\langle expr \rangle$ *is called* admissible *(with respect to the variable* n*), if it is constructed according to the following rules.*

(1) (built-in) Every expression free of n *(constants), the expression* n *itself (identity), every expression of the form* a^n *(exponential) with* a *free of* n*, and the expression* $n!$ *(factorial) is admissible.*

The following expressions are admissible:
BesselI$[n, x]$, BesselJ$[n, x]$, BesselK$[n, x]$, BesselY$[n, x]$,
Binomial$[\alpha n + \beta, \gamma n + \delta]$, ChebyshevT$[n, x]$, ChebyshevU$[n, x]$
Fibonacci$[n]$, Fibonacci$[n, x]$, Gamma$[n]$, GegenbauerC$[n, a, x]$,
HarmonicNumber$[n]$, HarmonicNumber$[n, r]$, HermiteH$[n, x]$,
JacobiP$[n, a, b, x]$, LaguerreL$[n, a, x]$, LegendreP$[n, x]$, Lucas$[n]$,
Lucas$[n, x]$, Pell$[n]$, Pell$[n, x]$, PellLucas$[n]$, PellLucas$[n, x]$,
RaisingFactorial$[n, d]$, FallingFactorial$[n, d]$
(a, b, x, γ, δ *free of* n*;* $d, r \in \mathbb{N}$*;* $\alpha, \beta \in \mathbb{Z}$*).*

(2) (user-defined) The expression $f[n]$*, where* f *is declared using the Where option (see below) is admissible.*

(3) (arithmetic) If $\langle expr \rangle_1, \langle expr \rangle_2$ *are admissible with respect to* n*, then so are*

$$\langle expr \rangle_1 + \langle expr \rangle_2, \ \langle expr \rangle_1 - \langle expr \rangle_2, \ \langle expr \rangle_1 \cdot \langle expr \rangle_2, \ \langle expr \rangle_1 / \langle expr \rangle_2.$$

In the latter case, it is assumed implicitly that the sequence corresponding to $\langle expr \rangle_2$ *does not vanish in the domain of definition.*

For $a \in \mathbb{Z}$*,* ${\langle expr \rangle_1}^a$ *is admissible.*

(4) (quantifiers) If $\langle expr \rangle$ *is admissible with respect to* i *and free of* n*, then the expressions* SUM$[\langle expr \rangle, \{i, a, n\}]$ *and* PRODUCT$[\langle expr \rangle, \{i, a, n\}]$ *are admissible in* n *for any* $a \in \mathbb{Z}$*.*

If $\langle expr \rangle_1, \langle expr \rangle_2$ *are admissible with respect to* i *and free of* n*, then the expressions*

$$\text{CFRAC}[\langle expr \rangle_1, \{i, a, n\}] \quad \textit{and} \quad \text{CFRAC}[\langle expr \rangle_2, \langle expr \rangle_1, \{i, a, n\}]$$

are admissible with respect to n *for any* $a \in \mathbb{Z}$*. These expressions correspond to the sequences of (partial) continued fractions*

$$\mathop{\mathrm{K}}_{i=a}^{n}(g(i), f(i)) := f(a) + \cfrac{g(a+1)}{f(a+1) + \cfrac{g(a+2)}{\cdots + \cfrac{g(n)}{f(n)}}}$$

where $f(n)$ *and* $g(n)$ *are the sequences corresponding to* $\langle expr \rangle_1, \langle expr \rangle_2$*, respectively. If* $\langle expr \rangle_2$ *is not specified, it is assumed that* $g(n) = 1$ *for all* n*.*

(5) (affine transforms) If $\langle expr \rangle$ *is admissible with respect to* n *and* $\langle expr \rangle'$

is obtained from $\langle expr \rangle$ by replacing each n with $an + b$ for some fixed $a, b \in \mathbb{N}_0$, then $\langle expr \rangle'$ is admissible.

If $\langle expr \rangle'$ is obtained from $\langle expr \rangle$ by replacing each n with Floor$[pn+q]$ *for some fixed $p, q \in \mathbb{Q}$, $p, q \geq 0$, then $\langle expr \rangle'$ is admissible.*

This rule may not be applied twice in a row, i.e., nested floor expressions like $\lfloor q \lfloor un + v \rfloor + q \rfloor$ are currently not allowed.

(6) (C-finite nesting) Expressions of the form $f[\langle expr \rangle]$ are admissible if f is specified by a linear homogeneous recurrence with constant coefficients (also called a C-finite recurrence), and if $\langle expr \rangle$ is an expression that corresponds to a sequence which satisfies a linear homogeneous recurrence with integer coefficients.

The inner expression $\langle expr \rangle$ must belong to the closure of constants, n, exponentials a^n (a free of n), and expressions $g[an + b]$ ($a, b \in \mathbb{Z}$, g user-defined or built-in) under addition, multiplication, exponentiation with a positive integer, and indefinite summation.

Sums and products are represented by the symbols SUM and PRODUCT in order to avoid conflicts with the symbols Sum and Product that have a predefined meaning in Mathematica.

For some admissible expressions, it is necessary to specify additional information in order to clarify which admissible sequence they are supposed to mean. Such supplementary information can be specified via options. In particular, using the Where option, sequences can be specified by an explicit admissible system given as a list of equations as specified in Definition 1 and equations of the form $f[i] == y$ with $i \in \mathbb{Z}$ for specifying initial values. The right hand side of the recurrence equation may well involve other admissible expressions as coefficients of the rational functions.

The variable in an admissible expression need not be n, it can be any Mathematica expression which is atomic with respect to the rules of Definition 2. SumCracker tries to automatically detect what the variable is, but it may fail if there are several plausible choices. In this case, the option Variable can be used for clarification.

We have introduced sequences as functions $f\colon \mathbb{N} \to k$. More generally, we regard any function $f\colon \{n_0, n_0 + 1, n_0 + 2, \dots\} \to k$ for some fixed startpoint $n_0 \in \mathbb{Z}$ as a sequence. Given an admissible expression, SumCracker assumes as startpoint the least number n_0 for which all sequences in the admissible system are defined (according to the specified initial values). If it is impossible to determine a startpoint from the initial values, then the default startpoint 0 is chosen. This may happen if the expression at hand only consists of built-in expressions like Fibonacci$[n]$, which are defined for all integers $n \in \mathbb{Z}$. The automatic detection of the startpoint can be bypassed by specifying the startpoint directly using the option From.

4 Proving Identities and Inequalities

4.1 Identities: ZeroSequenceQ

The proving command ZeroSequenceQ decides for an admissible expression whether it represents the zero sequence. In order to prove an identity $A = B$, this command is applied to the difference $A - B$. The identity holds if and only if the command returns True upon this input.

For instance, the q-Cassini identity

$$d_n e_{n+1} - d_{n+1} e_n = (-1)^n q^{\binom{n}{2}} \qquad (n \geq 0)$$

due to Andrews et. al. [5], where

$$\begin{aligned} d_{n+2} &= d_{n+1} + q^n d_n, \quad d_0 = 1, d_1 = 0, \\ e_{n+2} &= e_{n+1} + q^n e_n, \quad e_0 = 0, e_1 = 1, \end{aligned}$$

is easily established as follows:

In[11]:= $\mathbf{ZeroSequenceQ}[\boldsymbol{d[n]e[n+1] - d[n+1]e[n] - (-1)^n q^{\mathrm{Binomial}[n,2]}},$
$\quad \mathbf{Where} \to \boldsymbol{\{d[n+2] == d[n+1] + q^n d[n],}$
$\quad\quad \boldsymbol{d[0] == 1, d[1] == 0,}$
$\quad\quad \boldsymbol{e[n+2] == e[n+1] + q^n e[n],}$
$\quad\quad \boldsymbol{e[0] == 0, e[1] == 1\}]}$

Out[11]= True

Also identities involving "arbitrary sequences" can be proven. An example for this kind of identities is the Christoffel-Darboux identity for orthogonal polynomials [6, Thm. 4.5]. For arbitrary sequences c_n and λ_n, let the sequence $p_n(x)$ be defined via

$$p_n(x) = (x - c_n)p_{n-1}(x) - \lambda_n p_{n-2}(x) \quad (n \geq 0), \qquad p_{-1}(x) = 0, p_0(x) = 1.$$

Then:

$$\sum_{k=0}^{n} \frac{p_k(x)p_k(u)}{\prod_{i=1}^{k+1} \lambda_i} = \frac{p_{n+1}(x)p_n(u) - p_n(x)p_{n+1}(u)}{(x-u)\prod_{k=1}^{n+1} \lambda_n},$$

and we can prove this identity automatically for arbitrary c_n and λ_n. The Free option is used for specifying that the function symbols c and λ should denote free sequences.

In[12]:= $\mathbf{ZeroSequenceQ}[$
$\quad \mathbf{SUM}[\boldsymbol{px[k]pu[k]}/\mathbf{PRODUCT}[\boldsymbol{\lambda[i], \{i, 1, k+1\}}], \boldsymbol{\{k, 0, n\}}]$
$\quad - \dfrac{\boldsymbol{px[n+1]pu[n] - px[n]pu[n+1]}}{\boldsymbol{(x-u)}\mathbf{PRODUCT}[\boldsymbol{\lambda[i], \{i, 1, n+1\}}]},$
$\quad \mathbf{Where} \to \boldsymbol{\{px[n+2] == (x - c[n+2])px[n+1] - \lambda[n+2]px[n],}$
$\quad\quad \boldsymbol{px[-1] == 0, px[0] == 1,}$

$$\begin{aligned}&\boldsymbol{pu[n+2] == (u-c[n+2])pu[n+1]-\lambda[n+2]pu[n],}\\&\boldsymbol{pu[-1] == 0, pu[0] == 1\},}\end{aligned}$$

Free $\rightarrow$ $\boldsymbol{\{c,\lambda\}}$]

Out[12]= True

A more difficult example is the continued fraction identity

$$(a_0 - x) + \cfrac{xa_0}{(a_1 - x) + \cfrac{xa_1}{\cdots + \cfrac{xa_{n-1}}{a_n - x}}} = \frac{1}{\sum_{k=0}^{n} (-x)^k / \prod_{i=0}^{k} a_i} - x,$$

which holds for any sequence a in $\mathbb{C}(x) \setminus \{0, x\}$ [7]. Also this identity can be proven automatically in full generality:

In[13]:= $\mathbf{ZeroSequenceQ[CFRAC[}\boldsymbol{a[k-1]x, a[k]-x, \{k,0,n\}] + x}$
$\boldsymbol{- 1/}\mathbf{SUM}\boldsymbol{[(-x)^k/}\mathbf{PRODUCT}\boldsymbol{[a[i], \{i,0,k\}], \{k,0,n\}],}$
Free $\rightarrow$ $\boldsymbol{\{a\}}$]

Out[13]= True (*5.56s)

For this example, the built-in Gröbner basis facilities of Mathematica are not efficient enough. In order to obtain the result, we have outsourced all Gröbner basis computations to the special purpose system Singular.

4.2 Inequalities: ProveInequality

There is also a command by which some combinatorial inequalities can be proven. The command ProveInequality accepts an inequality of the form $A\diamondsuit B$ with $\diamondsuit \in \{=, \neq, \geq, \leq, >, <\}$ and returns True or False depending on whether the formula $A\diamondsuit B$ holds or not. Unlike ZeroSequenceQ, the inequality prover might not terminate.

As an example, consider the inequality

$$\sum_{k=1}^{n} \frac{L_k^2}{F_k} \geq \frac{(L_{n+2} - 3)^2}{F_{n+2} - 1} \quad (n \geq 2),$$

proposed by Diaz and Egozcue [10], where F_k is the k-th Fibonacci number and L_k denotes the k-th Lucas number, defined by

$$L_{k+2} = L_k + L_{k+1} \quad (k \geq 0), \qquad L_0 = 2, L_1 = 1.$$

In[14]:= **ProveInequality[SUM[Lucas[**$\boldsymbol{k}$**]**$^{\mathbf{2}}$**/Fibonacci[**$\boldsymbol{k}$**],** $\boldsymbol{\{k,1,n\}}$]
$\geq$ **(Lucas[**$\boldsymbol{n+2}$**]** $-$ **3)**$^{\mathbf{2}}$**/(Fibonacci[**$\boldsymbol{n+2}$**]** $-$ **1),**

From → 2]

This runs longer than the patience of the user permits. Probably it does not terminate at all. In such situations, termination can often be obtained by specifying some additional knowledge using the Using option. In this examples, it suffices to supply the fact $F_n \geq 1$ for all $n \in \mathbb{N}$. If desired, such additional information can afterwards be proven by the same procedure.

Out[14]= $Aborted (> 10h)

In[15]:= $\mathbf{ProveInequality[SUM[Lucas[}k\mathbf{]^2/Fibonacci[}k\mathbf{], \{}k\mathbf{, 1, }n\mathbf{\}]}$
$\mathbf{\geq (Lucas[}n\mathbf{ + 2] - 3)^2/(Fibonacci[}n\mathbf{ + 2] - 1),}$
$\mathbf{From \to 2, Using \to \{Fibonacci[}n\mathbf{] \geq 1\}]}$

Out[15]= True

In[16]:= $\mathbf{ProveInequality[Fibonacci[}n\mathbf{] \geq 1, From \to 2]}$

Out[16]= True

A lot of classical inequalities can be proven by this procedure. One example is Bernoulli's inequality.

In[17]:= $\mathbf{ProveInequality[(1 + }x\mathbf{)^}n\mathbf{ \geq 1 + }n\,x\mathbf{, Using \to \{}x\mathbf{ \geq -1\}]}$

SumCracker::general : Unable to detect variable. There are several equally reasonable possibilities.

Out[17]= $Failed

In[18]:= $\mathbf{ProveInequality[(1 + }x\mathbf{)^}n\mathbf{ \geq 1 + }n\,x\mathbf{,}$
$\mathbf{Variable \to }n\mathbf{, Using \to \{}x\mathbf{ \geq -1\}]}$

Out[18]= True

Observe here that the Variable option has to be used to prevent SumCracker from choosing x as the discrete variable. Also observe that the Using option was used here to specify the domain of the parameter x. Most textbook authors overlook that Bernoulli's inequality already holds from $x = -2$ on:

In[19]:= $\mathbf{ProveInequality[(1 + }x\mathbf{)^}n\mathbf{ \geq 1 + }n\,x\mathbf{,}$
$\mathbf{Variable \to }n\mathbf{, Using \to \{}x\mathbf{ \geq -2\}]}$

Out[19]= True

ProveInequality also supports the Free option for specifying "arbitrary sequences." For example, the Cauchy-Schwarz inequality can be proved automatically this way.

According to our experience, the ProveInequality command does not terminate for inequalities of outstanding difficulty such as the inequalities of Vietoris or

Askey-Gasper [4, Chapter 7], and for those it is also not possible to obtain termination by adding some trivial additional knowledge. However, the procedure successfully applies to many elementary inequalities which are easy but perhaps cumbersome to prove by hand. It might be useful for proving inequalities which are not of interest in their own right, but which appear as subproblems in the proof of more sophisticated theorems.

The most nontrivial inequality we know on which ProveInequality succeeds is Turan's inequality

$$P_{n+1}(x)^2 - P_n(x)P_{n+2}(x) \geq 0 \qquad (-1 \leq x \leq 1)$$

for Legendre polynomials [35,12]:

In[20]:= $\mathbf{ProveInequality[}$
$\mathbf{LegendreP}[\boldsymbol{n}+\mathbf{1},\boldsymbol{x}]^{\mathbf{2}} - \mathbf{LegendreP}[\boldsymbol{n},\boldsymbol{x}]\mathbf{LegendreP}[\boldsymbol{n}+\mathbf{2},\boldsymbol{x}] \geq \mathbf{0},$
$\mathbf{Using} \to \{-\mathbf{1} \leq \boldsymbol{x} \leq \mathbf{1}\}, \mathbf{Variable} \to \boldsymbol{n}]$

Out[20]= True (3.74s)

Analogous inequalities also hold (and can be proven) for other families of polynomials, such has Hermite polynomials, Laguerre polynomials, etc.

5 "Cracking" Expressions: Crack

The Crack command has already been introduced in Section 2. It takes an admissible expression $f(n)$ as input and attempts to find an expression which is simpler than the original one. To be more precise, Crack searches for a multivariate rational function r such that

$$f(n) = r(f_1(n), \ldots, f_m(n)),$$

where $f_1(n), \ldots, f_m(n)$ are automatically determined from the subexpressions of $f(n)$. Alternatively, the user can also specify $f_1(n), \ldots, f_m(n)$ explicitly by using the Into option.

5.1 Indefinite Summation

In indefinite summation, the goal is to eliminate the outermost summation quantifier from an expression of the form $F(n) := \sum_{k=1}^{n} f(k)$. Typical examples include identities for orthogonal polynomials such as [4, Chapter 6]

$$\sum_{k=0}^{n} (k+\lambda)C_k^\lambda(x) = \frac{(n+2\lambda)C_n^\lambda(x) - (n+1)C_{n+1}^\lambda(x)}{2(1-x)}.$$

In[21]:= $\mathbf{Crack[SUM[(\boldsymbol{k}+\boldsymbol{\lambda})GegenbauerC[\boldsymbol{k},\boldsymbol{\lambda},\boldsymbol{x}], \{\boldsymbol{k},0,\boldsymbol{n}\}]]}$

Out[21]=
$$\frac{(n+2\lambda)C_n^\lambda(x)-(n+1)C_{n+1}^\lambda(x)}{2(1-x)}$$

Section 2 above contains further examples. In general, it might not be possible to simplify a given sum. If this is the case, then the original expression is returned:

In[22]:= $\mathbf{Crack[SUM[1/Lucas[2^{\boldsymbol{k}}], \{\boldsymbol{k}, 0, \boldsymbol{n}\}, From \to 1]}$

Out[22]=
$$\sum_{k=0}^{n}\frac{1}{L_{2^k}}$$

This output means that SumCracker was not able to find a simpler representation for the sum $\sum_{k=0}^{n} 1/L_{2^k}$. This does not mean, however, that no closed form for the sum exists at all, it only means that there is no closed form in the search space that SumCracker has investigated. If Crack exceeds a certain heuristically chosen total degree bound for numerator and denominator of the rational function r on the right hand side, then it gives up and returns the sum unsimplified. The degree bound can be specified explicitly using the Degree option. By choosing a degree bound, beware that the runtime of the underlying algorithm is exponential in the degree and the number of subexpressions of the sum in the worst case. For the sum $\sum_{k=0}^{n} 1/L_{2^k}$, this worst case complexity is not attained, so we can raise the degree quite far:

In[23]:= $\mathbf{Crack[SUM[1/Lucas[2^{\boldsymbol{k}}], \{\boldsymbol{k}, 0, \boldsymbol{n}\}, From \to 1, Degree \to 10]}$

Out[23]=
$$\sum_{k=0}^{n}\frac{1}{L_{2^k}}$$

In[24]:= $\mathbf{Crack[SUM[1/Lucas[2^{\boldsymbol{k}}], \{\boldsymbol{k}, 0, \boldsymbol{n}\}, From \to 1, Degree \to 250]}$

Out[24]=
$$\sum_{k=0}^{n}\frac{1}{L_{2^k}} \tag{5.6h}$$

Now we can be sure that if there does exist closed form of $\sum_{k=0}^{n} 1/L_{2^k}$ in terms of a rational function in L_{2^k} and L_{2^k+1} then this rational function must have a numerator or a denominator with total degree more than 250.

5.2 Simplification

The Crack command is not restricted to indefinite sums. It can be applied to any admissible expression, and thus can also be used as a simplifier for admissible expressions.

As a simple example, consider the Perrin sequence p_n [33, A001608], defined via

$$p_{n+3} = p_n + p_{n+1}, \quad p_0 = 3, p_1 = 0, p_2 = 2.$$

This sequence give rise to a simple pseudo primarity test [3]. Expressions like

$$\frac{p_{n+2}p_{n+1}^3 - p_{n+1}^4 - p_{n+2}^3 p_{n+1} + 23p_{n+1}}{p_n^2 + 2p_{n+1}p_n + p_{n+1}^2 - p_{n+2}^2 - 3p_{n+1}p_{n+2}}$$

involving p_n can be simplified by Crack:

In[25]:= $\mathbf{Crack[(p[n+2]p[n+1]^3 - p[n+1]^4 - p[n+2]^3 p[n+1] + 23p[n+1])/}$
$\mathbf{(p[n]^2 + 2p[n+1]p[n] + p[n+1]^2 - p[n+2]^2 - 3p[n+1]p[n+2]),}$
$\mathbf{Where \rightarrow \{p[n+3] == p[n] + p[n+1],}$
$\mathbf{p[0] == 3, p[1] == 0, p[2] == 2\}]}$

Out[25]=
$$p_n p_{n+1}$$

5.3 Conversion

By default, the Crack command determines the expressions $f_1(n), \ldots, f_m(n)$ which might appear in the output from the subexpressions of the input. The choice of the $f_i(n)$ can also be done explicitly by the user, using the Into option. In connection with this option, the Crack command resembles the convert function of Maple.

As a simple example, we might want to eliminate the shift in a from the Jacobi polynomial $P_n^{(a+1,b)}(x)$. We can do this by typing the following command line:

In[26]:= $\mathbf{Crack[JacobiP[n, a+1, b, x], Into \rightarrow \{n, JacobiP[n, a, b, x]\}]}$

Out[26]=
$$\frac{2(n+1)P_{n+1}^{(a,b)}(x) - 2(1+a+n)P_n^{(a,b)}(x)}{(2n+a+b+2)(x-1)} \tag{21.20s}$$

This result coincides with (22.7.15) of [2]. The embarrassingly long runtime in this example is caused by the presents of the three parameters a, b, x. For a quicker example, let $d(n)$ be the number of paths in an $n \times n$ grid from the south-west corner to the north-east corner, using only single steps north, east, north-east [33, A001850, Delannoy numbers], and let $s(n)$ be the number of paths of the same type, which do not touch any point above the diagonal from south-west to north-east [33, A006318, Schröder numbers]. It can be shown [8] that these numbers satisfy the recurrences

$$d(n+2) = \frac{3(2n+3)d(n+1) - (n+1)d(n)}{n+2}, \quad d(1) = 3, d(2) = 13,$$
$$s(n+2) = \frac{3(2n+3)s(n+1) - ns(n)}{n+3}, \qquad s(1) = 2, s(2) = 6.$$

If we want to express, say, $n^2 s(n) + s(n+1)/n$ in terms of the Delannoy numbers, we can type the following.

In[27]:= $\mathbf{Crack}[n^2 s[n] + s[n+1]/n, \mathbf{Into} \to \{n, d[n]\},$

$\mathbf{Where} \to \{d[n+2] == \dfrac{3(2n+3)d[n+1] - (n+1)d[n]}{2+n},$

$d[1] == 3, d[2] == 13,$

$s[n+2] == \dfrac{3(2n+3)s[n+1] - n\,s[n]}{3+n},$

$s[1] == 2, s[2] == 6\}]$

Out[27]=
$$\frac{(n^3+2n^2+3)d(n+1) - (3n^3+6n^2+1)d(n)}{2n(n+2)} \tag{3.89s}$$

5.4 Solving Nonlinear Difference Equations

Crack is also useful for solving certain nonlinear difference equations. As a simple example, the difference equation

$$u(n+1) = \frac{3u(n)+1}{5u(n)+3} \quad (n \geq 1), \qquad u(1) = 1$$

has been posed by Rabinowitz [27]. A solution in terms of Fibonacci numbers is requested.

We can solve this problem by regarding the difference equation as a definition for the unknown function u and applying Crack to express this u in terms of the expressions that we expect in the solution.

In[28]:= $\mathbf{Crack}[u[n], \mathbf{Into} \to \{\mathbf{Fibonacci}[n]\},$

$\mathbf{Where} \to \{u[n+1] == (3u[n]+1)/(5u[n]+3), u[1] == 1\}]$

Out[28]=
$$\frac{-2F_n^2 + 2F_nF_{n+1} - F_{n+1}^2}{4F_n^2 - 6F_nF_{n+1} + F_{n+1}^2}$$

In Section 6.3 below, we will show how SumCracker can be used to automatically generate problems of this kind.

6 Discovering Algebraic Dependencies: ApproximateAnnihilator

The Crack command described in the previous section is a specialized form of the more general command ApproximateAnnihilator. This command can be used for discovering algebraic dependencies among admissible sequences. An algebraic dependency among sequences $f_1, \ldots, f_m \colon \mathbb{N} \to k$ is a polynomial $p \in k[x_1, \ldots, x_m]$ such that

$$p(f_1(n), \ldots, f_m(n)) = 0 \qquad (n \in \mathbb{N}).$$

The set of all algebraic dependencies forms an ideal in $k[x_1,\dots,x_m]$, which we call the *annihilator* of $f_1,\dots,f_m$. (This ideal must, however, not be confused with an ideal of annihilating linear difference operators, as used, e.g., in the work of Zeilberger [37]).

Observe that the results of a call Crack[$\langle f_1\rangle$, Into $\to$ $\{\langle f_2\rangle,\dots,\langle f_m\rangle\}$] are nothing else but algebraic dependencies of the special shape

$$p(f_2(n),\dots,f_m(n))f_1(n) - q(f_2(n),\dots,f_m(n)) = 0 \qquad (n \geq 0).$$

In fact, the same algorithm is used for Crack and ApproximateAnnihilator. The only difference in the implementation is that in Crack the search is restricted to dependencies of the above form, so that this command runs usually faster than the general command.

The general command ApproximateAnnihilator takes a list $\{f_1(n),\dots,f_m(n)\}$ of admissible expressions and a symbol x as input and returns a (Gröbner) basis of the ideal generated by all algebraic dependencies $p \in k[x_1,\dots,x_m]$ of a prescribed total degree. The default degree bound 10 can be overruled by the option Degree.

Some situations where this command is helpful are described in order.

6.1 q-Cassini's Identity

In Section 4 we have shown how the q-analogue

$$d_n e_{n+1} - d_{n+1} e_n = (-1)^n q^{\binom{n}{2}}$$

of Cassini's identity can be proven automatically. If we want to find such an identity, the Crack command is of little help.

Of course, we could find the identity via

In[29]:= $\boldsymbol{def = \{d[n+2] == d[n+1] + q^n d[n], d[0] == 1, d[1] == 0,}$
$\boldsymbol{e[n+2] == e[n+1] + q^n e[n], e[0] == 0, e[1] == 1\};}$

In[30]:= $\mathbf{Crack}[\boldsymbol{d[n]e[n+1] - d[n+1]e[n]}, \mathbf{Into} \to \{(-1)^{\boldsymbol{n}}, \boldsymbol{q}^{\mathbf{Binomial}[\boldsymbol{n},2]}\},$
$\mathbf{Where} \to \boldsymbol{def}, \mathbf{Variable} \to \boldsymbol{n}]$

Out[30]= $$(-1)^n q^{\binom{n}{2}},$$

but this requires knowing that we have to crack the left hand side into $(-1)^n$ and $q^{\binom{n}{2}}$. If we do not know this, we can blindly search for algebraic dependencies between the entities $d_n, d_{n+1}, e_n, e_{n+1}, (-1)^n$ and q^n.

In[31]:= $\mathbf{ApproximateAnnihilator}[\{\boldsymbol{d[n], d[n+1], e[n], e[n+1], (-1)^n, q^n}\}, \boldsymbol{x},$
$\mathbf{Where} \to \boldsymbol{def}, \mathbf{Degree} \to \mathbf{5}]$

Out[31]= $$\{x_5^2 - 1\} \qquad (32.01\text{s})$$

This gives just the dependency $\left((-1)^n\right)^2 = 1$. Next we might include q^{n^2} into the search.

In[32]:= **ApproximateAnnihilator[**
{d[n], d[n + 1], e[n], e[n + 1], (−1)^n, q^n, q^{n^2}}, x,
Where → def, Degree → 5, Variable → n]

Out[32]= $\{x_5^2 - 1, x_2^2 x_3^2 x_6 - 2x_1 x_2 x_3 x_4 x_6 + x_1^2 x_4^2 x_6 - x_7\}$ (135.46s)

The second dependency gives corresponds to the identity

$$(d_n e_{n+1} - d_{n+1} e_n)^2 q^n = q^{n^2},$$

hence

$$|d_n e_{n+1} - d_{n+1} e_n| = q^{(n^2-n)/2} = q^{\binom{n}{2}}.$$

Now by considering initial values, it is easily seen that

$$(d_n e_{n+1} - d_{n+1} e_n)/q^{\binom{n}{2}} = (-1)^n,$$

from which the desired identity follows.

In[33]:= **Clear[def];**

6.2 Somos Sequences

A Somos sequence [34,11] of order r is a sequence C_n which satisfies a recurrence equation of the form

$$C_{n+r}C_n = C_{n+r-1}C_{n+1} + C_{n+r-2}C_{n+2} + \cdots + C_{n+r-\lfloor r/2\rfloor}C_{n+\lfloor r/2\rfloor}. \qquad (2)$$

It can be shown that when $C_0, C_1, \ldots, C_{r-1}$ are nonzero integral initial values, then C_n is a nonzero integer for every $n \in \mathbb{N}$, which in particular means that the sequence C_n is well defined by initial values and the difference equation above. Upon division by C_n it becomes apparent that C_n is an admissible sequences.

It is of interest [36] to know whether a given Somos sequence of order r is also a Somos sequence of some different order r'. SumCracker supports investigations of this kind. To be specific, let C_n be defined by

$$C_{n+4} = (C_{n+3}C_{n+1} + C_{n+2}^2)/C_n \quad (n \geq 4), \qquad C_0 = C_1 = C_2 = C_3 = 1.$$

In order to find out whether this sequence also satisfies equations of the form (2) for $r \neq 4$, we will search for corresponding polynomials in the ideal of algebraic dependencies.

In[34]:= ***vars* = {C[n], C[n + 1], C[n + 2], C[n + 3], C[n + 4], C[n + 5],**
C[n + 6], C[n + 7]};

```
In[35]:= id = ApproximateAnnihilator[vars,
        Where → {C[n + 4] == (C[n + 3]C[n + 1] + C[n + 2]^2)/C[n],
            C[0] == 1, C[1] == 1, C[2] == 1, C[3] == 1},
        Degree → 2];                                              (20.30s)
In[36]:= id = GroebnerBasis[id, vars];
```

We search for the desired polynomials by reducing an ansatz polynomial modulo the ideal *id* and comparing the coefficients of the obtained normal form to zero. The solutions of the resulting linear system are precisely the required polynomials. (Note that the restriction Degree → 2 is well justified for our purpose.)

```
In[37]:= ansatz[r_] :=
        Sum[a[i]C[n + r − i]C[n + i], {i, 0, Floor[r/2]}] /. a[0] → 1;
In[38]:= FindSomos[r_] := ansatz[r] /.
        First[
          Solve[
            Thread[
              CoefficientList[
                Last[
                  PolynomialReduce[ansatz[r], id, vars]
        ], vars] == 0]]];
In[39]:= FindSomos[4]
```

Out[39]=

$$C_{n+4}C_n - C_{n+3}C_{n+1} - C_{n+2}^2$$

In[40]:= FindSomos[5]

Out[40]=

$$C_{n+5}C_n + C_{n+4}C_{n+1} - 5C_{n+3}C_{n+2}$$

In[41]:= FindSomos[6]

Solve::svars : Equations may not give solutions for all "solve" variables.

Out[41]=

$$C_{n+6}C_n - (a_3 + 5)C_{n+5}C_{n+1} + (a_3 + 4)C_{n+4}C_{n+2} + a_3C_{n+3}^2$$

In[42]:= % /. {{a[3] → 0}, {a[3] → 1}}

Out[42]=

$$\{C_{n+6}C_n - 5C_{n+5}C_{n+1} + 4C_{n+4}C_{n+2},\; C_{n+6}C_n - 6C_{n+5}C_{n+1} + 5C_{n+4}C_{n+2} + C_{n+3}^2\}$$

In[43]:= FindSomos[7] /. {{a[3] → 0}, {a[3] → 1}}

Solve::svars : Equations may not give solutions for all "solve" variables.

Out[43]=

$$\{C_{n+7}C_n - \tfrac{4}{5}C_{n+6}C_{n+1} - \tfrac{29}{5}C_{n+5}C_{n+2},\; C_{n+7}C_n - C_{n+6}C_{n+1} - 6C_{n+5}C_{n+2} + C_{n+4}C_{n+3}\}$$

This list is exhaustive in the sense that every other Somos-like relation of C_n of order at most 7 is a linear combination of those which appear above as output. By leaving the coefficients in the recurrence and the initial values symbolic, we found that that every sequence C_n satisfying

$$C_{n+4} = (\alpha C_{n+3}C_{n+1} + \beta C_{n+2}^2)/C_n$$

also satisfies

$$\begin{aligned}
C_{n+5}C_n &= \frac{\beta\left(\beta C_1^2+\alpha C_0C_2\right)C_2^2+\beta C_0^2C_3^2+\alpha C_1\left(\beta C_1^2+\alpha C_0C_2\right)C_3}{C_0C_1C_2C_3}C_{n+3}C_{n+2} \\
&\quad - \beta C_{n+4}C_{n+1}, \\
C_{n+6}C_n &= \Big(\frac{\beta^3-\alpha^4}{\beta} - \frac{\left(\alpha C_3C_1^3+\beta C_2^2C_1^2+C_0\left(\alpha C_2^3+C_0C_3^2\right)\right)\alpha^2}{C_0C_1C_2C_3}\Big)C_{n+4}C_{n+2} \\
&\quad + \Big(\frac{\left(\beta C_1^2+\alpha C_0C_2\right)\alpha^2}{\beta C_0C_2} + \frac{C_0C_3\alpha}{C_1C_2} + \frac{\alpha\beta C_2C_1^2+\alpha^2C_0C_2^2}{C_0C_1C_3}\Big)C_{n+5}C_{n+1}, \\
C_{n+7}C_n &= \Big(\frac{\beta C_0C_3\alpha^2}{C_1C_2} + \frac{\beta C_2\left(\beta C_1^2+\alpha C_0C_2\right)\alpha^2}{C_0C_1C_3} \\
&\qquad + \frac{C_0C_2\alpha^4+\beta C_1^2\alpha^3-\beta^3C_0C_2}{C_0C_2}\Big)C_{n+4}C_{n+3} \\
&\quad + \Big(\frac{C_0C_3\beta^2}{C_1C_2} + \frac{C_2\left(\beta C_1^2+\alpha C_0C_2\right)\beta^2}{C_0C_1C_3} \\
&\qquad + \frac{\alpha\left(\beta C_1^2+\alpha C_0C_2\right)\beta}{C_0C_2}\Big)C_{n+5}C_{n+2},
\end{aligned}$$

etc.

In[44]:= **ClearAll[*vars*, *id*, *ansatz*, *FindSomos*];**

6.3 Automatically Posing Quarterly Problems

Many relationships which can be found in the problem sections of contemporary mathematical journals are consequences of algebraic dependencies. In the previous section, we have found with the Crack command a sequence $u(n)$ satisfying the recurrence

$$u(n+1) = \frac{3u(n)+1}{5u(n)+3}, \quad u(1) = 1.$$

Conversely, we may use ApproximateAnnihilator to design equations like this for prescribed solutions, for instance the solution $u(n) = F_{3n}/L_{3n}$:

In[45]:= **ApproximateAnnihilator[{$\frac{\text{Fibonacci}[3n]}{\text{Lucas}[3n]}$, $\frac{\text{Fibonacci}[3(n+1)]}{\text{Lucas}[3(n+1)]}$}, *u*, Degree → 2]**

Out[45]= $\{-1 - 2u[1] + 2u[2] + 5u[1]u[2]\}$

It follows that the desired equation reads

$$u(n+1) = \frac{2u(n)+1}{5u(n)+2}, \quad u(1) = 1.$$

More generally, for any $u(n) = a(n)/b(n)$ where both $a(n)$ and $b(n)$ satisfy the same recurrence of order two with constant coefficients, such an equation can be found.

In[46]:= **ApproximateAnnihilator[{a[n]/b[n], a[n + 1]/b[n + 1]}, u,
Where → {a[n + 2] == x · a[n + 1] + y · a[n],
b[n + 2] == x · b[n + 1] + y · b[n]},
Degree → 2]**

Out[46]= $\{ya_1^2 + xa_2a_1 - yb_1u[1]a_1 - yb_1u[2]a_1 - xb_2u[2]a_1 - a_2^2 - xa_2b_1$
$u[1] + a_2b_2u[1] + a_2b_2u(2) + yb_1^2u[1]u[2] - b_2^2u[1]u[2] + xb_1b_2u[1]u[2]\}$

Cleaning up this output leads to the equation

$$u(n+1) = \frac{(a_2b_2 - ya_1b_1 - xa_2b_1)\,u(n) + ya_1^2 + xa_2a_1 - a_2^2}{(b_2^2 - yb_1^2 - xb_2b_1)\,u(n) + ya_1b_1 + xa_1b_2 - a_2b_2}, \quad u(1) = \frac{a_1}{b_1}.$$

These relationships can also be obtained easily from the theory of continued fractions [25]. The point here is that no knowledge of this theory is required if the SumCracker package is used.

In[47]:= **Quit**

7 Conclusion

SumCracker is a software package for dealing with nonstandard expressions involving symbolic sums and, more generally, recursively defined sequences. It provides tools for proving identities and inequalities, and for finding identities of a prescribed form. A collection of example problems was given which at present cannot be solved by any other software package.

Some examples, however, could well be done by hand, at least by people who have some experience in the manipulation of special sequences. The choice of these examples was forced by the extreme runtime complexity that prevented harder examples from going through. Improving the efficiency of the package is thus a major objective for the development of future versions. For instance, the general procedures currently implemented in the package could be combined with classical "special purpose" summation algorithms, which are usually much faster and thus should be used whenever possible.

It would of course also be interesting to develop more powerful algorithms which apply to the whole class of admissible sequences. For instance, some of the work that Schneider [30,31, e.g.] has undertaken for the case of ΠΣfields might be transferable to some extend. Currently under development is an extension to definite summation problems, i.e., to sums $\sum_{k=0}^{n} f(n,k)$ where the summand and summation bound need not be independent. At the moment, SumCracker does not support this kind of sums.

Finally, we would like to point out that the algorithms underlying the SumCracker package admit differential analogues (except for the inequality prover). In the definition of admissibility, the ith shift $f(n+i)$ just has to be exchanged by the ith derivative $f^{(i)}(z)$. For instance, the Lambert W function [9], defined as the solution $W(z)$ of the equation $z = w\exp(w)$ satisfies

$$W'(z) = \frac{W(z)}{1+W(z)}$$

and hence is admissible in this sense. An implementation of these algorithms could be of interest as well.

References

[1] S.A. Abramov, J.J. Carette, K.O. Geddes, and H.Q. Le. Telescoping in the context of symbolic summation in Maple. *Journal of Symbolic Computation*, 38(4):1303–1326, 2004.

[2] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, Inc., 9 edition, 1972.

[3] W. Adams and D. Shanks. Strong primality tests that are not sufficient. *Mathematics of Computation*, 39:255–300, 1982.

[4] George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*, volume 71 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, 1999.

[5] George E. Andrews, Arnold Knopfmacher, and Peter Paule. An infinite family of Engel expansions of Rogers-Ramanujan type. *Advances in Applied Mathematics*, 25:2–11, 2000.

[6] Theodore S. Chihara. *An Introduction to Orthogonal Polynomials*, volume 13 of *Mathematics and its Applications.* Gordon and Breach Science Publishers, 1978.

[7] G. Chrystal. *Algebra — an Elementary Textbook, Part II.* Cambridge University Press, 2nd edition, 1922.

[8] Louis Comtet. *Advanced Combinatorics.* D. Reidel Publishing Company, 1970.

[9] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the lambert W function. *Adv. Comput. Math.*, 5, 1996.

[10] José Luis Diaz and Juan J. Egozcue. Problem B-943. *The Fibonacci Quarterly*, 40(4):372, 2002.

[11] David Gale. The strange and surprising saga of the Somos sequences. *Mathematical Intelligencer*, 13(1):40–42, 1991.

[12] Stefan Gerhold and Manuel Kauers. A computer proof of Turán's inequality. Technical Report 2005-15, SFB F013, Johannes Kepler Universität, 2005. (submitted).

[13] Stefan Gerhold and Manuel Kauers. A procedure for proving special function inequalities involving a discrete parameter. In *Proceedings of ISSAC'05*, pages 156–162, 2005.

[14] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, second edition, 1994.

[15] Gert-Martin Greuel and Gerhald Pfister. *A Singular Introduction to Commutative Algebra*. Springer, 2002.

[16] Michael Karr. Summation in finite terms. *Journal of the ACM*, 28:305–350, 1981.

[17] Michael Karr. Theory of summation in finite terms. *Journal of Symbolic Computation*, 1(3):303–315, 1985.

[18] Manuel Kauers. An algorithm for deciding zero equivalence of nested polynomially recurrent sequences. Technical Report 2003-48, SFB F013, Johannes Kepler Universität, 2003. (submitted).

[19] Manuel Kauers. Computer proofs for polynomial identities in arbitrary many variables. In *Proceedings of ISSAC '04*, pages 199–204, July 2004.

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

[21] Manuel Kauers. Solving difference equations in difference rings (???). Technical report, SFB F013, Johannes Kepler Universität, 2005. (in preparation).

[22] Edouard Lucas. Théorie des fonctions numérique simplement périodiques. *American Journal of Mathematics*, 1:184–240, 289–321, 1878.

[23] D. A. Millin. Problem H-237. *The Fibonacci Quarterly*, 12(3):309, 1974.

[24] Peter Paule and Markus Schorn. A Mathematica version of Zeilberger's algorithm for proving binomial coefficient identities. *Journal of Symbolic Computation*, 20(5–6):673–698, 1995.

[25] Oskar Perron. *Die Lehre von den Kettenbrüchen*. Chelsea Publishing Company, second edition, 1929.

[26] Marko Petkovšek, Herbert Wilf, and Doron Zeilberger. $A = B$. AK Peters, Ltd., 1997.

[27] Stanley Rabinowitz. Problem B-951. *The Fibonacci Quarterly*, 40(1):84, 2003.

[28] Axel Riese. qMultiSum — a package for proving q-hypergeometric multiple summation identities. *Journal of Symbolic Computation*, 35:349–376, 2003.

[29] Carsten Schneider. *Symbolic Summation in Difference Fields.* PhD thesis, RISC-Linz, Johannes Kepler Universität Linz, 2001.

[30] Carsten Schneider. Symbolic summation with single-nested sum extensions. In *Proceedings of ISSAC '04*, pages 282–289, July 2004.

[31] Carsten Schneider. Finding telescopers with minimal depth for indefinite nested sum and product expressions. In *Proceedings of ISSAC '05*, pages 285–292, July 2005.

[32] Andrew V. Sills. RRtools — a Maple package for aiding the discovery and proof of finite Rogers–Ramanujan type identities. *Journal of Symbolic Computation*, 37(4):415–448, 2004.

[33] Neil J.A. Sloane. The on-line encyclopedia of integer sequences. http://research.att.com/˜njas/sequences/.

[34] Michael Somos. Problem 1470. *Crux Mathematicorum*, 15:208, 1989.

[35] Gabor Szegő. On an inequality of P. Turán concerning Legendre polynomials. *Bulletin of the American Math. Society*, 54:401–405, 1948.

[36] Alf van der Poorten. Elliptic curves and continued fractions. *Journal of Integer Sequences*, 8(2):1–19, 2005.

[37] Doron Zeilberger. A holonomic systems approach to special functions. *Journal of Computational and Applied Mathematics*, 32:321–368, 1990.