THE MECHANIZATION OF THE PROOFS OF COMBINATORIAL IDENTITIES

by

Alessia M Venter

B.S., Mathematics
Wheaton College

2001

A thesis submitted to the
University of Colorado at Denver
and Health Sciences Center
in partial fulfillment
of the requirements for the degree of
Master of Science
Applied Mathematics

2006
This thesis for the Master of Science
degree by
Alexsis M Venter
has been approved
by

[Signatures]

Bill Cherowitzo
Stan Payne
Michael Jacobson

April 17, 2006
Date
Venter, Alexis M (M.S., Applied Mathematics)
The Mechanization of the Proofs of Combinatorial Identities
Thesis directed by Professor Bill Cherowitzo

ABSTRACT

For hypergeometric sums, the transition has been made from hand written proofs to mechanized computer proofs. The non-mechanized algorithm of Sister Mary Celine Fasenmyer led to a foundational theorem stating that any proper hypergeometric sum satisfies a recurrence relation of the form

$$\sum_{i=0}^{l} \sum_{j=0}^{j} a_{i,j}(n)F(n-j, k-i) = 0.$$  

Gosper’s algorithm (for indefinite sums) and the Zeilberger-Petkovsek algorithm (for definite sums) are mechanized procedures for finding such recurrences and subsequently stating the closed form representation of the original sum. These algorithms together give a complete solution to the question of whether a sum of proper hypergeometric terms can be written in "closed form." They will either find the desired identity or let you know that the sum cannot be represented in such a way. Moreover, the WZ algorithm provides a certificate which can be used for manual verification.

This abstract accurately represents the content of the candidate’s thesis.
recommend its publication.

Signed  
Bill Cherowitzo
DEDICATION

This thesis is dedicated first and foremost to my husband,
Jaco Venter, and also to

my family:
Byron & Donna Chapin
Koos & Louisa Venter
Albert Preston
Gwen Janssen
Orpa Aylward
Chapins, Prestons, Gardners,
Janssens, Venters, and Camphers

and all my former math teachers at:
Francis Howell North High School, St. Charles, Missouri
Wheaton College, Wheaton, Illinois
University of Colorado, Denver, Colorado
ACKNOWLEDGMENT

This thesis would not have been possible without the encouragement and guidance of Bill Cherowitzo, to whom I am extremely grateful.
## CONTENTS

1. Introduction .................................................. 1
2. Preliminaries .................................................. 3
   2.1 Generating Functions ........................................ 3
   2.2 Hypergeometric Sums ........................................ 3
   2.3 Canonical Forms ............................................ 4
   2.4 Resultants .................................................. 5
   2.5 Closed Form ............................................... 7
3. Past Methods .................................................. 8
   3.1 The Snake Oil Method ...................................... 8
   3.2 The Hypergeometric Database .............................. 11
3.3 Sister Mary Celine Fasanmyer’s Algorithm .................. 13
3.4 Brief Analysis of Mary Celine’s Algorithm .................. 16
4. Gosper’s Algorithm ......................................... 20
   4.1 The Algorithm .............................................. 20
   4.2 Examples of Gosper’s Algorithm .......................... 25
   4.3 Analysis of Gosper’s Algorithm ........................... 30
5. Zeilberger’s Algorithm ...................................... 33
   5.1 The Algorithm .............................................. 33
   5.2 An example of Zeilberger’s Algorithm ...................... 36
   5.3 Analysis of Zeilberger’s algorithm ........................ 38
6. WZ Algorithm ........................................... 40
6.1 The Algorithm ........................................ 40
6.2 An Example of the WZ Algorithm .................... 42
6.3 Analysis of the WZ Algorithm ....................... 43
7. Algorithm “Hyper” ..................................... 47
7.1 “Poly” ................................................. 47
7.2 “Hyper” ............................................... 49
7.3 Examples Using Algorithms Poly and Hyper ....... 50
8. Summary ................................................. 54
References ................................................. 55
1. Introduction

Over the past century, the degree to which computers are involved in the world around us has increased explosively. Now we are not only finding more ways to make use of them, but also smarter ways. Computer programming together with mathematical algorithms has saved much time and energy in performing mathematical computations. In other fields as well, we are learning to delegate more tasks to the computer, and in effect pushing our cognitive skills out of their comfort zones and into new realms.

One instance of this ever more common phenomenon is the use of computers in finding identities of combinatorial sums. Through recent history, many mathematicians have spent much time solving individual identities in which they found particular interest or relevance. And the need for these identities will continue to be of great importance. As solving these identities could seem an infinite project, a sharp algorithm or any sort of procedure for minimizing the task would be highly valued. Mathematicians have formed methods and algorithms for specific classes of sums to minimize the work. And recently, algorithms have been developed and refined to do this automatically by computer.

Automated theorem proving, or ATP, is the proving of mathematical theorems by a computer program. Depending on the underlying logic, the problem of deciding the validity of a theorem varies from trivial to impossible. A related problem is proof verification, where an existing proof for a theorem is certified valid. For this, it is generally required that each individual proof step can be
verified by a primitive recursive function or program, and hence the problem is always decidable. *Interactive theorem provers* require a human user to give hints to the system. Depending on the degree of automation, the prover can essentially be reduced to a proof checker, with the user providing the proof in a formal way, or significant proof tasks can be performed automatically. Interactive provers are used for a variety of tasks, but even fully automatic systems have by now proven a number of interesting and hard theorems, including some that have eluded human mathematicians for a long time. However, these successes are sporadic, and work on hard problems usually requires a proficient user. A *computer-assisted proof* is a mathematical proof that has been generated by computer [18].

There is much to benefit from this transition of hand worked proofs to computer solutions. This paper will give a survey of the past and current methods and algorithms for combinatorial identities and where we stand in the solution of these problems.
2. Preliminaries

2.1 Generating Functions

A basic knowledge of generating functions is required for understanding the Snake Oil Method in Section 3.1.

A generating function is a formal power series

\[ f(x) = \sum_{n=0}^{\infty} a_n x^n \]

whose coefficients give the sequence \( \{a_0, a_1, \ldots\} \) [13].

2.2 Hypergeometric Sums

A hypergeometric series \( \sum_k t_k \) is a series for which \( t_0 = 1 \) and the consecutive term ratio \( \frac{t_{k+1}}{t_k} \) is a rational function of the summation index \( k \) [14].

Another way of viewing this is that

\[ \frac{t_{k+1}}{t_k} = \frac{P(k)}{Q(k)} \]

where \( P(k) \) and \( Q(k) \) are polynomial functions.

In plain terms, for further illumination, a hypergeometric sum will have a summand including only factorials, polynomials, and exponential functions of the summation variable.

Example 1. Let

\[ f(n) = \sum_k \binom{n}{k} \frac{k!}{3^k} . \]

Then \( t_0 = \binom{n}{0} = 1 = 1 \), and the consecutive term ratio is

\[ \frac{t_{k+1}}{t_k} = \frac{n! k!(n-k)!}{3(k+1)!(n-k-1)!n!} = \frac{n-k}{3(k+1)} . \]
where clearly the numerator and denominator are both rational functions of $k$.

**Example 2.** Let

$$f(n) = \sum_k \sin\left(\frac{k\pi}{6}\right).$$

Then $t_0 = 0$, and the consecutive term ratio is

$$\frac{\sin\left(\frac{(k+1)\pi}{6}\right)}{\sin\left(\frac{k\pi}{6}\right)},$$

where there is no clear simplification, and neither the numerator or denominator is a rational function of $k$.

### 2.3 Canonical Forms

A canonical form is a clear-cut way of describing every object in a class in a one-to-one manner [15]. A canonical form is required to have two essential properties. Every object under consideration must have exactly one canonical form, and two objects that have the same canonical form must be essentially the same [16]. Then to verify that some entity $A$ is equal to an entity $B$, you must just check to see that the canonical form of $A$ is equal to the canonical form of $B$.

**Example 3.** Verify that $3(x - 4)^2 + 9x = 3(16 - x + x^2 - 4x)$.

Write each polynomial in order of descending powers for your canonical form and you get

$$3(x^2 - 8x + 16) + 9x = 48 - 3x + 3x^2 - 12x.$$  
$$3x^2 - 24x + 48 + 9x = 48 - 15x + 3x^2.$$  
$$3x^2 - 15x + 48 = 3x^2 - 15x + 48.$$
For a common, non-mathematical example, a Social Security Number would (ideally) be a good canonical form to verify that two statements regarding a U.S. Citizen refer to the same person.

2.4 Resultants

In step 2 of Gosper's Algorithm, we need the idea of the Resultant of two polynomials. Given $p$ and $q$, two polynomials which factor into linear factors

$$p(x) = a_0(x - r_1)(x - r_2) \cdots (x - r_m)$$

$$q(x) = b_0(x - s_1)(x - s_2) \cdots (x - s_n),$$

the resultant is

$$R(p(x), q(x)) = a_0^m b_0^n \prod_{i=1}^{m} \prod_{j=1}^{n} (r_i - s_j).$$

**Example 4.** Let $p(x) = (x - h)^2$ and $q(x) = 4(x + 2)(x - 7)(x + 5)$. Then $a_0 = 1, b_0 = 4, m = 2, n = 3$ and $r_1 = h, r_2 = h, s_1 = -2, s_2 = 7, s_3 = -5$. Then the Resultant of $p$ and $q$ is

$$R((x - h)^2, 4(x + 2)(x - 7)(x + 5)) = 1^3 4^2 \prod_{i=1}^{2} \prod_{j=1}^{3} (r_i - s_j).$$

$$R(p, q) = 16 \left[ (h + 2)(h - 7)(h + 5)(h + 2)(h - 7)(h + 5) \right].$$

$$R(p, q) = 16(h + 2)^2(h - 7)^2(h + 5)^2.$$
\[ q(x) = \sum_{i=0}^{n} b_ix^{n-i}. \]

Then the resultant can be expressed by the following \( m+n \) by \( m+n \) determinant:

\[
\begin{vmatrix}
  a_0 & a_1 & a_2 & \ldots & a_m & 0 & \ldots & 0 \\
  0 & a_0 & a_1 & \ldots & a_{m-1} & a_m & \ldots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \ldots & \ldots & a_m & & & & \\
  b_0 & b_1 & b_2 & \ldots & b_n & 0 & \ldots & 0 \\
  0 & b_0 & b_1 & \ldots & b_{n-1} & b_n & \ldots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & b_n \\
\end{vmatrix}
\]

**Example 5.** In the previous example, consider how we would use the determinant if the polynomials were unfactored: \( p(x) = x^2 - 2hx + h^2 \) and \( q(x) = 4x^3 - 156x - 280. \)

\[
\begin{vmatrix}
  1 & -2h & h^2 & 0 & 0 \\
  0 & 1 & -2h & h^2 & 0 \\
  0 & 0 & 1 & -2h & h^2 \\
  4 & 0 & -156 & -280 & 0 \\
  0 & 4 & 0 & -156 & -280 \\
\end{vmatrix}
\]

The above determinant is \( 16h^6 - 1248h^4 - 2240h^3 + 24336h^2 + 87360h + 78400 \) which is equivalent to the factored form \( 16(h + 2)^2(h - 7)^2(h + 5)^2. \)
2.5 Closed Form

Definition 2.1 For our purposes in dealing with hypergeometric terms and sums, a function $f(n)$ can be thought of as being in closed form if it is equal to a linear combination of a fixed number, say $r$, of hypergeometric terms. The number $r$ must be an absolute constant, i.e., it must be independent of all variables and parameters of the problem.
3. Past Methods

Many methods have been developed to take care of certain classes of sums. Many of these are clever and quite helpful, but also now obsolete.

3.1 The Snake Oil Method

One slick procedure is the Snake Oil method. The idea here is to use generating functions to find whole classes of sums and pluck the sum in question from the coefficients of the generating function. The steps from Wilf [7] are as follows:

1. Identify the free variable, say $n$, that the sum depends on. Give a name to the sum that you are working on; call it $f(n)$.

2. Let $F(x)$ be the ordinary power series generating function ("opsgf") whose $[x^n]$ is $f(n)$, the sum that you’d love to evaluate. The notation $[x^n]$ denotes the coefficient of $x^n$ in any generating function. In this setting, basically we have,

$$F(x) \equiv \sum_n f(n)x^n.$$

3. Multiply the sum by $x^n$ and sum over $n$. Your generating function is now expressed as a double sum, over $n$ and over whatever variable was first used as a dummy summation variable.

4. Interchange the order of the two summations that you are now looking at, and express the inner one in simple closed form. For this purpose it will
be helpful to have a catalogue of series whose sums are known, such as the
list in section 2.5 of [7].

5. Try to identify the coefficients of the generating function of the answer,
because those coefficients are what you want to find.

**Example 6.** Use the Snake Oil method to find a closed form representation for

\[
\sum_{k\geq 0} \binom{k}{n-k} \quad (n = 0, 1, 2, \ldots).
\]

For step 1, we note that the free variable is \( n \) and \( k \) is the dummy variable. Let

\[
f(n) = \sum_{k\geq 0} \binom{k}{n-k}.
\]

Then \( F(x) = \sum_n f(n)x^n \) is the generating function with which we are concerned.

Proceeding with the method, we multiply by \( x^n \) and sum over \( n \).

\[
F(x) = \sum_n f(n)x^n = \sum_n \sum_{k\geq 0} \binom{k}{n-k} x^n.
\]

Next we interchange the summations and express the inner sum in closed form. This is the step in which we need a catalogue or listing of series whose sums are known. This is a significant downfall of the Snake Oil method.

\[
F'(x) = \sum_{k\geq 0} \sum_n \binom{k}{n-k} x^n.
\]

\[
F(x) = \sum_{k\geq 0} x^k \sum_n \binom{k}{n-k} x^{n-k}.
\]
Now let r,n-k and we have a convenient conversion for the inner sum.

\[ F(x) = \sum_{k \geq 0} x^k \sum_{r} \binom{k}{r} x^r. \]

\[ F(x) = \sum_{k \geq 0} x^k (1 + x)^k. \]

\[ F(x) = \sum_{k \geq 0} (x + x^2)^k. \]

Then, via knowledge of generating functions, we know that

\[ F(x) = \frac{1}{1 - (x + x^2)} = \frac{1}{1 - x - x^2}. \]

This generating function is known to generate the Fibonacci numbers giving

\[ f(n) = F_n \text{ where } F_{-1} = 0, F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 3, F_4 = 5, \text{ etc.} \]

\[ \sum_{k \geq 0} \binom{k}{n - k} = F_n \quad (n = 0, 1, 2, \ldots). \]

The closed form we have for \(F_n\) is an approximation.

\[ F_n \approx \frac{1}{\sqrt{5}} \left( \frac{1 + \sqrt{5}}{2} \right)^{n+1} \]

Many sums that are commonly solved with the Snake Oil method are hypergeometric sums. From this point forward, we consider methods dealing only with hypergeometric sums.
3.2 The Hypergeometric Database

As previously mentioned, much time has been spent on solving individual sums. One problem with this is that much time can be spent solving equivalent sums which are disguised to look differently. A step toward eliminating this problem is the formation of a canonical form for hypergeometric sums. Any hypergeometric sum can be converted into this canonical form, and then we can "look it up" in the hypergeometric database.

1. Given a series $\sum_k l_k$. Shift the summation index $k$ so that the sum starts at $k = 0$ with a nonzero term. Extract the term corresponding to $k = 0$ as a common factor so that the first term of the sum will be 1.

2. Simplify the ratio $\frac{t_{k+1}}{t_k}$ to bring it into the form $\frac{P(k)}{Q(k)}$, where $P, Q$ are polynomials. If this cannot be done, the series is not hypergeometric.

3. Completely factor the polynomials $P$ and $Q$ into linear factors, and write the term ratio in the form

$$\frac{P(k)}{Q(k)} = \frac{(k + a_1)(k + a_2)\ldots(k + a_p)}{(k + b_1)(k + b_2)\ldots(k + b_q)(k + 1)^x}.$$ 

If the factor $k + 1$ in the denominator wasn't there, put it in, and compensate by inserting an extra factor of $k + 1$ in the numerator. Notice that all of the coefficients of $k$, in numerator and denominator, are $+1$. Whatever numerical factors are needed to achieve this are absorbed into the factor $x$. 

11
4. You have now identified the input series. It is the common factor that you
extracted in step 1 above, multiplied by the hypergeometric series which
is defined by:

\[ \begin{bmatrix} a_1 & a_2 & \cdots & a_p \\ \vdots \end{bmatrix}_{x}^{\begin{bmatrix} b_1 & b_2 & \cdots & b_q \end{bmatrix}}.\]

**Example 7.** Use the Hypergeometric Database look up method to find a
closed form representation of the sum

\[ \sum_{k} \binom{n}{k} x^k.\]

We already know the closed form for this sum, but we will use this simple example to illustrate the method.

For step 1, note that when \( k = 0 \), the summand is nonzero. Also, the term corresponding to \( k = 0 \) is 1, so we need not extract anything.

The consecutive term ratio is

\[ \frac{t_{k+1}}{t_k} = \frac{\binom{n}{k+1} x^{k+1}}{\binom{n}{k} x^k} = \frac{n-k}{k+1} x = \frac{k-n}{k+1} (-x).\]

Thus we have

\[ \begin{bmatrix} -n \\ \vdots \\ - \end{bmatrix}_{x}^{\begin{bmatrix} - \end{bmatrix}}.\]

We now look up \( \begin{bmatrix} -n \\ \vdots \\ - \end{bmatrix} \) in our small list of identities in chapter 3 of [7] and find

\[ \begin{bmatrix} a \\ \vdots \\ - \end{bmatrix}_{x}^{\begin{bmatrix} a \\ \vdots \\ - \end{bmatrix}} = \frac{1}{(1-z)^a}.\]

Letting \( a = -n \) and \( z = -x \)
\[ _1F_0 \left[ \begin{array}{c} -n \\ _- \\ - \end{array} ; -x \right] = \frac{1}{(1 + x)^{-n}} = (1 + x)^n. \]

Clearly a huge drawback is that we must reference some list or database of known identities. There may be any number of partial lists; you can create one for yourself easily. The problem is that no comprehensive hypergeometric database exists as the number of possible identities is infinite.

### 3.3 Sister Mary Celine Fasenmyer’s Algorithm

We now consider an algorithm which unlike the previous two methods is free of dependence on lists and catalogues of known identities.

We are given a sum, \( f(n) = \sum_k F(n, k) \), where \( F \) is doubly hypergeometric. That is to say both the ratio, \( \frac{F(n, k+1)}{F(n, k)} \), and the ratio \( \frac{F(n+1, k)}{F(n, k)} \) are hypergeometric. Ultimately, we are looking for a recurrence formula for the sum, \( f(n) \), and we start by finding a recurrence for the summand, \( F(n, k) \). Our recurrence for the summand will be in the form

\[
\sum_{i=0}^{I} \sum_{j=0}^{J} a_{i,j}(n) F(n - j, k - i) = 0. \tag{3.1}
\]

The steps of the algorithm are as follows.

1. Fix trial values of \( I \) and \( J \), say \( I = J = 1 \).

2. Assume the recurrence formula in the form of (3.1) with the coefficients \( a_{i,j}(n) \) to be determined, if possible.

3. Divide each term of (3.1) by \( F(n, k) \) and reduce each ratio \( \frac{F(n-j, k-i)}{F(n, k)} \) by simplifying the ratios of the factorials that it contains, so that only rational functions of \( n \) and \( k \) remain.
4. Place the entire expression over a single common denominator. Then collect the numerator as a polynomial in \( k \).

5. Solve the system of linear equations that results from equating to zero the coefficients of each power of \( k \) in the numerator polynomial, for the unknown coefficients of \( a_{i,j} \). If the system has no solution, try the whole thing again with larger values of \( I \) and/or \( J \). That is, look for a bigger recurrence.

6. Once the recurrence for \( F(n, k) \) has been determined, sum said recurrence over \( k \) and work with the result to find the recurrence for the sum, \( f(n) \).

**Example 8.** Use Sister Celine’s algorithm to find a closed form of the sum

\[
f(n) = \sum_k \binom{n}{k}.
\]

Again, it is no challenge to find this sum - we know its closed form by heart. We will use this simple sum to demonstrate the steps of Mary Celine’s algorithm.

Let \( I = J = 1 \).

\[
\sum_{i=0}^{1} \sum_{j=0}^{1} a_{i,j}(n)F(n-j,k-i) = 0. 
\] (3.2)

\[
a_{0,0}(n)F(n,k)+a_{0,1}(n)F(n-1,k)+a_{1,0}(n)F(n,k-1)+a_{1,1}(n)F(n-1,k-1) = 0.
\]

Change notation so that the coefficients are most simple and don’t need subscripts. Remember that they do still depend on \( n \).

\[
aF(n,k) + bF(n-1,k) + cF(n,k-1) + dF(n-1,k-1) = 0. 
\] (3.3)

\[
a\binom{n}{k} + b\binom{n-1}{k} + c\binom{n}{k-1} + d\binom{n-1}{k-1} = 0.
\]
\[ a \frac{n!}{k!(n-k)!} + b \frac{(n-1)!}{k!(n-1-k)!} + c \frac{n!}{(k-1)!(n-k+1)!} + d \frac{(n-1)!}{(k-1)!(n-k)!} = 0. \]

Now divide through by \( F(n, k) = \frac{n!}{k!(n-k)!} \).

\[ a \frac{n!(n-k)!}{k!(n-1-k)!n!} + b \frac{(n-1)!(n-k)!}{k!(n-1-k)!n!} + c \frac{n!(n-k)!}{(k-1)!(n-k+1)!} + d \frac{(n-1)!(n-k)!}{(k-1)!(n-k)!n!} = 0. \]

\[ a + b \frac{(n-1)!(n-k)!}{(n-1-k)!n!} + c \frac{k!(n-k)!}{(k-1)!(n-k+1)!} + d \frac{(n-1)!(n-k)!}{(k-1)!(n-k)!n!} = 0. \]

\[ a + b \frac{n-k}{n} + c \frac{k}{n-k+1} + d \frac{k}{n} = 0. \]

Next, find a common denominator and collect terms in the numerator as a polynomial in \( k \).

\[ a \frac{n(n-k+1)}{n(n-k+1)} + b \frac{(n-k)(n-k+1)}{n(n-k+1)} + c \frac{nk}{n(n-k+1)} + d \frac{k(n-k+1)}{n(n-k+1)} = 0. \]

\[ \frac{an(n-k+1) + b(n-k)(n-k+1) + cnk + dk(n-k+1)}{n(n-k+1)} = 0. \]

\[ \frac{k^2(b-d) + k(-an - 2bn - b + cn + dn + d) + (an^2 + an + bn^2 + bn)}{n(n-k+1)} = 0. \]

Thus we have the following system of equations

\[ b - d = 0 \] (3.4)

\[ -an - 2bn - b + cn + dn + d = 0 \] (3.5)

\[ an^2 + an + bn^2 + bn = 0 \] (3.6)

which yield the solution set \( d(-1, 1, 0, 1) \).

Use \( d = 1 \) and equation (3.3) to obtain the resulting recurrence for \( F(n, k) \):

\[ -F(n, k) + F(n-1, k) + F(n-1, k-1) = 0. \]

\[ F(n, k) = F(n-1, k) + F(n-1, k-1). \]
(Note that this recurrence is what we would expect to find based on Pascal’s Triangle.)

Now sum this recurrence over all integer values of \( k \). If \( k < 0 \) or \( k > n \), the summand \( F(n, k) \) will vanish and so the sums contain only finite numbers of terms. We get that

\[
\sum_{k} F(n, k) = \sum_{k} F(n - 1, k) + \sum_{k} F(n - 1, k - 1).
\]

\[
f(n) = f(n - 1) + f(n - 1).
\]

\[
f(n) = 2f(n - 1).
\]

Using some small computed values \([f(0) = 1, f(1) = 2, f(2) = 4]\), and our simplified recurrence, we determine that

\[
f(n) = 2^n.
\]

### 3.4 Brief Analysis of Mary Celine’s Algorithm

Although Sister Celine’s method has been much improved upon, it retains its importance as being one of the first results regarding computerized proofs of identities. Also, her algorithm led to a general existence theorem for hypergeometric sums’ recurrence relations which we will now state.

**Definition 3.1** A function \( F(n, k) \) is said to be a proper hypergeometric term if it can be written in the form

\[
F(n, k) = P(n, k)\prod_{i=1}^{m}(a_i n + b_i k + c_i)! \prod_{i=1}^{n}(u_i n + v_i k + w_i)! x^k,
\]

in which \( x \) is an indeterminate over, say, the complex numbers, and
1. $P$ is a polynomial,

2. the $a$'s, $b$'s, $u$'s, $v$'s are specific integers, that is to say, they do not contain any additional parameters, and

3. the quantities $uu$ and $uv$ are finite, nonnegative, specific integers[7].

**Theorem 3.2** Let $F(n, k)$ be a proper hypergeometric term. Then $F$ satisfies a $k$-free recurrence relation. That is to say, there exist positive integers $I, J$, and polynomials $a_{i,j}(n)$ for $i = 0, \ldots, I; j = 0, \ldots, J$, not all zero, such that the recurrence

$$
\sum_{i=0}^{I} \sum_{j=0}^{J} a_{i,j}(n) F(n-j,k-i) = 0. \quad (3.7)
$$

holds at every point $(n,k)$ at which $F(n,k) \neq 0$ and all of the values of $F$ that occur in (3.7) are well defined. Furthermore, there is such a recurrence with $(I, J) = (I^*, J^*)$ where

$$
J^* = \sum_s |b_s| + \sum_s |v_s|; I^* = 1 + \text{deg}(P) + J^* \left( \left\{ \sum_s |a_s| + \sum_s |u_s| \right\} - 1 \right).
$$

This theorem guarantees that Sister Celine's algorithm will succeed if $I$ and $J$ are large enough. It also provides specific values for which the algorithm will work. These values, $I^*$ and $J^*$, are not necessarily the smallest such values that will work.

**Proof:** A lengthy proof of the first part of this theorem is given by Wilf and Zeilberger in [12]. Before beginning the proof, preliminary work is done to write the generic ratio, $\frac{F(n-\delta_{i,j}k-i)}{F(n,k)}$, as

$$
\frac{F(n-j,k-i)}{F(n,k)} = \frac{v(n,k)}{\delta(n,k)}, \quad (3.8)
$$
where $v(n, k)$ equals
\[
P(n-j, k-i) = \prod_{a=1}^{n} \prod_{b=1}^{m} r_f([a_{a,j} + b_{s,i}], a_{a} n + b_{s} k + c_s) \prod_{u_{a,j} + v_{s,i} < 0} f_f(u_{a,j} + v_{s,i}, u_{a} n + v_{s} k + w_s),
\]
and $\delta(n, k)$ equals
\[
P(n,k) = \prod_{a=1}^{n} \prod_{b=1}^{m} f_f(a_{a,j} + b_{s,i}, a_{a} n + b_{s} k + c_s) \prod_{u_{a,j} + v_{s,i} < 0} r_f([u_{a,j} + v_{s,i}], u_{a} n + v_{s} k + w_s).
\]

The notation for rising factorial and falling factorial are as follows: $r_f(x, y) = \prod_{j=1}^{x} (y+j)$ and $f_f(x, y) = \prod_{j=0}^{x-1} (y-j)$. All other notation is as in Definition 3.1. To begin, assume the recurrence (3.7), and try to solve for the coefficients, $a_{i,j}$, by first dividing through by $P(n, k)$. Then we have
\[
\sum_{0 \leq i \leq I, 0 \leq j \leq J} a_{i,j}(n) \frac{v_{i,j}(n, k)}{\delta_{i,j}(n, k)} = 0. \tag{3.9}
\]

Next a common denominator is found to divide
\[
P(n, k) \prod_{a=1}^{n} \prod_{b=1}^{m} f_f(a_{a}^{+} + b_{s}^{+} + I, a_{a} n + b_{s} k + c_s) \prod_{u_{a,j} + v_{s,i} < 0} r_f((-u_{a})^{+} + (-v_{s})^{+} + I, u_{a} n + v_{s} k + w_s), \tag{3.10}
\]
where $x^{+} = \max(x, 0)$. The fractions are then cleared out by multiplying (3.9) through by (3.10). We get
\[
\sum_{0 \leq i \leq I, 0 \leq j \leq J} a_{i,j}(n) v_{i,j}(n, k) \frac{\Delta}{\delta_{i,j}(n, k)} = 0, \tag{3.11}
\]
a polynomial in $k$, where $\Delta$ is the common denominator (3.10). Finally the coefficients of each power of $k$ in (3.11) are equated to zero. In order to validate this theorem, we must verify that the system of linear equations resulting from (3.11) has a nontrivial solution. Due to the properties of rising and falling factorials, the degree of $k$ grows linearly with $I$ and $J$. The number of unknowns, $a_{i,j}$, grows similarly to $IJ$ (it is actually $(I+1)(J+1)$). Then when $I$ and $J$ are
sufficiently large, the number of unknowns will surpass the number of equations in our system, and we will be guaranteed a nontrivial solution. An additional proof is omitted that guarantees $I^*$ and $J^*$ are satisfactorily large values.

The statement of this theorem is foundational to the mechanized proofs we consider next. It is strong — it says we can find a recurrence relation for any proper hypergeometric term. The algorithms we consider next give specific steps for finding this recurrence and then for finding a closed form representation from the recurrence.
4. Gosper’s Algorithm

Gosper’s algorithm is of great importance in the computerization of closed form summation problems. Not only does it completely solve what it sets out to do, but Gosper’s algorithm plays a foundational role in both Zeilberger’s Creative Telescoping algorithm and the WZ algorithm. Gosper’s algorithm answers the following question: Given a hypergeometric term $t_n$, is there a hypergeometric term $z_n$ such that $z_{n+1} - z_n = t_n$?

If Gosper’s algorithm succeeds, then we know that the sum of the hypergeometric term can be written as another hypergeometric term, plus some constant. If Gosper’s algorithm fails, then we know that the sum cannot be written as a hypergeometric term plus a constant, nor can it be written as a linear combination of a fixed number of hypergeometric terms. (It cannot be written in our closed form.)

4.1 The Algorithm

Suppose we are given a sum

$$s_n = \sum_{k=0}^{n-1} t_k,$$

where $t_k$ is a hypergeometric term. If we could express $t_k$ as the difference of two hypergeometric terms,

$$t_k = z_{k+1} - z_k; \quad (4.1)$$

20
then we are led to a simple looking relationship.

\[ z_n = z_{n-1} + t_{n-1} = z_{n-2} + t_{n-2} + t_{n-1} = \ldots = z_0 + \sum_{k=0}^{n-1} t_k = s_n + z_0, \]

where \( z_0 \) is some constant.

Given a hypergeometric term, \( t_n \), Gosper’s algorithm will either produce the desired term \( z_n \) (and we say that \( t_n \) is Gosper-summable) or verify that no such hypergeometric \( z_n \) exists.

If \( z_n \) is a hypergeometric term satisfying (4.1), then \( \frac{z_n}{t_n} \) is a rational function of \( n \). Call this unknown function \( y(n) \) and rewrite as \( z_n = y(n)t_n \). Substitute this value of \( z_n \) into (4.1) and we get

\[ r(n)y(n+1) - y(n) = 1, \]

where \( r(n) = \frac{t_{n+1}}{t_n} \). Now we have shifted the problem from finding hypergeometric solutions of (4.1) to finding rational solutions of (4.2), a linear recurrence.

Next, the problem is again shifted into finding polynomial solutions of another linear recurrence. Assume that \( r(n) \) can be factored in the following way:

\[ r(n) = \frac{a(n)c(n+1)}{b(n)c(n)}, \]

where \( a(n), b(n), \) and \( c(n) \) are polynomials in \( n \), and

\[ \gcd(a(n), b(n+h)) = 1, \]

for all nonnegative integers \( h \). Gosper tells us to look for a solution of (4.2) in the form

\[ y(n) = \frac{b(n-1)x(n)}{c(n)}. \]
Notice that now we have a representation for \( g(n) \) and it depends on an \( x(n) \) to which we now turn our attention. Substituting (4.3) and (4.5) into (4.2) we get that \( x(n) \) will satisfy

\[
a(n)x(n + 1) - b(n - 1)x(n) = c(n).
\]  \hspace{1cm} (4.6)

Gosper did prove a theorem that guarantees \( x(n) \) is a polynomial in \( n \) [3]. From this point, we need only to find \( x(n) \) and then, substituting again, return the value

\[
z_n = \frac{b(n - 1)x(n)}{c(n)} t_n.
\]  \hspace{1cm} (4.7)

Obviously there are many details to consider in addition to this general flow of relevant equations. The main computational aspects of Gosper’s algorithm can be divided into the following four steps and many substeps.

1. Form the ratio \( r(n) = \frac{t_{n+1}}{t_n} \) from the given hypergeometric term, \( t_n \). \( r(n) \) is a rational function of \( n \).

2. Write \( r(n) = \frac{a(n)}{b(n)} \frac{c(n+1)}{c(n)} \) where \( a(n), b(n), c(n) \) are polynomials satisfying (4.4).

At times it may seem that we can just easily spot the appropriate values for \( a(n), b(n), \) and \( c(n) \). However there are specific detailed guidelines that must be followed in order to correctly proceed.

Let \( f(n) \) be the numerator of \( r(n) \), and let \( g(n) \) be the denominator of \( r(n) \). If \( \gcd(f(n), g(n + h)) = 1 \) for all nonnegative integers \( h \), we can take \( a(n) = f(n), b(n) = g(n), \) and \( c(n) = 1 \) for our factorization of \( r(n) \). If not, we must “factor out” the common piece by following these steps:
(a) Let \( r(n) = Z \frac{f(n)}{g(n)} \) where \( f \) and \( g \) are monic relatively prime polynomials, and \( Z \) is a constant;

(b) Let \( R(h) \) be assigned the value of \( \text{Resultant}_n(f(n), g(n + h)) \).

(c) Let \( S = \{ h_1, h_2, \ldots, h_N \} \) be the set of nonnegative integer zeros of \( R(h) \) \( (N \geq 0, 0 \leq h_1 < h_2 < \ldots < h_N) \).

(d) Let \( p_0(n) = f(n) \) and \( q_0(n) = g(n) \). For \( j = 1, 2, \ldots, N \) perform the following loop:

i. Assign to \( s_j(n) \) the value of \( \gcd(p_{j-1}(n), q_{j-1}(n + h_j)) \).

ii. Assign to \( p_j(n) \) the value of \( \frac{p_{j-1}(n)}{s_j(n)} \).

iii. Assign to \( q_j(n) \) the value of \( \frac{q_{j-1}(n)}{s_j(n-h_j)} \).

(e) Let \( a(n) = Zp_N(n), b(n) = q_N(n), \) and \( c(n) = \prod_{i=1}^{N} \prod_{j=1}^{h_i} s_i(n-j) \).

3. Find a nonzero polynomial solution of \( x(n) \) of (4.6), if one exists; otherwise return \( \sum_{k=0}^{n-1} t_k \) and stop.

Step 3 consists of two main parts. First, you find an upper bound on the degree of \( x(n) \) via Gosper’s algorithm and second, you find the polynomial \( x(n) \) which satisfies (4.6) above using the method of undetermined coefficients. Before listing the steps, we introduce some notation and terminology. Let \( \deg a(n) \) be the degree of polynomial \( a(n) \). Let \( \text{lc} a(n) \) be the leading coefficient of \( a(n) \). Let \( A \) be the coefficient of the term of degree \( \deg a(n) - 1 \) in \( a(n) \). Let \( B \) be the coefficient of the term of degree \( \deg b(n-1) - 1 \) in \( b(n-1) \).
(a) If $\deg a(n) \neq \deg b(n)$ or $\text{lcm } a(n) \neq \text{lcm } b(n)$ then
\[
D = \{\deg c(n) - \max\{\deg a(n), \deg b(n)\}\}.
\]

(b) If $\deg a(n) = \deg b(n)$ and $\text{lcm } a(n) = \text{lcm } b(n)$ then
\[
D = \left\{\deg c(n) - \deg a(n) + 1, \frac{(B - A)}{\text{lcm } a(n)}\right\}.
\]

(c) Based on the above two cases, $D$ will contain either one or two elements. Now remove any elements from $D$ that are not positive integers or zero. If $D$ is now empty, return "no nonzero polynomial solution" and stop. Otherwise the upper bound for the degree of $x(n)$ is the maximum element in $D$.

(d) Use the method of undetermined coefficients to find a nonzero polynomial solution $x(n)$ of (4.6). If none exists, return "no nonzero polynomial solution" and stop.

4. Return $z_n = \frac{b(n-1)x(n)}{c(n)} t_n$ and stop. Step 4 is quite straightforward now that we have found $x(n)$.

Note that if our sum is not Gosper-summable, the problem will occur at step 3, i.e., we will not be able to find the desired polynomial $x(n)$. If the sum is Gosper-summable, then once we find $z_n$ we easily have $s_n = z_n - z_0$. If the lower bound is some $c_0$ rather than zero, that is no problem, we simply put $s_n = z_n - z_0$. 
4.2 Examples of Gosper’s Algorithm

Example 9. Use Gosper’s algorithm to find a closed form representation of the sum

\[ S_m = \sum_{n=0}^{m} \frac{n^4 4^n}{\binom{2n}{n}}. \]

Step 1 is quite simple and straightforward. We form \( r(n) = \frac{t_n+1}{t_n} \) and simplify with algebra.

\[ r(n) = \frac{(n + 1)^4 4^{n+1} \binom{2n}{n}}{n^4 4^n \binom{2n+2}{n+1}} = \frac{(n + 1)^4 (n + 1)^2}{n^4 (2n + 2)(2n + 1)} = \frac{2(n + 1)^5}{n^4 (2n + 1)}. \]

Step 2 requires that we write \( r(n) \) in the form of (4.3) above. In this example, it might seem that a good idea would be to take \( a(n) = 2(n + 1), b(n) = 2n + 1 \) and \( c(n) = n^4 \). What we would be missing is the fact that both the numerator and denominator, \( f \) and \( g \), must be monic polynomials. Thus we are not quite finished reducing \( r(n) \).

\[ r(n) = \frac{(n + 1)^5}{n^4(n + \frac{1}{2})}. \]

Now consider \( \gcd(f(n), g(n + h)) \). We would like to know if this \( \gcd \) is equal to 1 for all nonnegative integers \( h \). If so, we would have our simple factorization and move on to step 3. However, a quick counterexample of \( h = 1 \) shows that the \( \gcd \) is not always 1. If \( h = 1 \), then

\[ \gcd((n + 1)^5, (n + h)^4 (n + h + \frac{1}{2})) = \gcd((n + 1)^5, (n + 1)^4 (n + \frac{3}{2})) = (n + 1)^4. \]

So we must follow the extended rules for step 2.

- \( r(n) = \frac{(n+1)^3}{n^2(n+\frac{1}{2})} \) so \( Z = 1, f(n) = (n + 1)^5 \), and \( g(n) = n^4(n + \frac{3}{2}) \).

25
• We must find $R(h)$, the Resultant of $f(n)$ and $g(n+h)$. Using the method provided in the preliminaries, $R(h) = (h - 1)^20(h - \frac{1}{2})^5$.

• $S = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$ and $|S| = 20 = N$.

• Let $p_0(n) = (n + 1)^5$ and $q_0(n) = n^4(n + \frac{1}{2})$. For $j = 1, 2, \ldots, 20$ perform the loop to assign values to $s_j(n)$, $p_j(n)$, and $q_j(n)$.

\[
\begin{align*}
    s_1(n) &= (n + 1)^4 & p_1(n) &= n + 1 & q_1(n) &= n + \frac{1}{2} \\
    s_2(n) &= 1 & p_2(n) &= n + 1 & q_2(n) &= n + \frac{1}{2} \\
    s_3(n) &= 1 & p_3(n) &= n + 1 & q_3(n) &= n + \frac{1}{2} \\
    \vdots & & \vdots & & \vdots \\
    s_{20}(n) &= 1 & p_{20}(n) &= n + 1 & q_{20}(n) &= n + \frac{1}{2}
\end{align*}
\]

• This loop gives the final results we need to determine that $a(n) = n + 1$, $b(n) = n + \frac{1}{2}$, and $c(n) = n^4$.

Now we know the form of the equation that we must use to find $x(n)$.

\[
(n + 1)x(n + 1) - (n - \frac{1}{2})x(n) = n^4.
\]

Before looking at the directions for step 3, we note the following: $\deg a(n) = \deg b(n) = 1$, $\deg c a(n) = \deg b(n) = 1$, $A = 1$, and $B = -\frac{1}{2}$. Step 3.a. does not hold, so 3.b. does. Thus

\[
D = \left\{ 4 - 1 + 1, \frac{1}{2} - \frac{1}{2} \right\} = \left\{ 4, -\frac{1}{2} \right\}.
\]

When we remove the numbers that are not positive integers or zero, what we have left is the integer 4. This will be our $d$, the upper bound on the degree of
$x(n)$. Now use the method of undetermined coefficients and solve the following equation.

$$(n + 1)(c_0 + c_1(n + 1) + c_2(n + 1)^2 + c_3(n + 1)^3 + c_4(n + 1)^4)
- (n - \frac{1}{2})(c_0 + c_1n + c_2n^2 + c_3n^3 + c_4n^4) = n^4.$$

Using Derive 6 for computation,

$$x(n) = \frac{2}{11} n^4 - \frac{40}{99} n^3 + \frac{40}{231} n^2 + \frac{52}{693} n - \frac{4}{231}.$$

Step 4 is to return the value of $z_n = \frac{b(n - 1)x(n)}{c(n)}$ which in our case is

$$z_n = \frac{(n - \frac{1}{2})4^n}{\binom{2n}{n}} x(n).$$

We leave $x(n)$ unexpanded for now. Gosper’s algorithm has succeeded as it claimed it would. All that is left for us to do is to use $z_n$ to find our actual sum. Since we want the sum of $m + 1$ terms, we get $S_m = s_{m+1} = z_{m+1} - z_0$.

$$S_m = \frac{(m + \frac{1}{2})4^{m+1}}{\binom{2m+2}{m+1}} x(m + 1) - \frac{(0 - \frac{1}{2})4^0}{\binom{0}{0}} x(0).$$

$$S_m = \frac{(m + \frac{1}{2})4^{m+1}}{\binom{2m+2}{m+1}} x(m + 1) + \frac{1}{2} x(0).$$

$$S_m = \frac{2(2m + 1)4^m}{\binom{2m+2}{m+1}} x(m + 1) - \frac{2}{231}.$$

$$S_m = \frac{2(2m + 1)4^m}{(2m+2)(2m+1)[2m+1]} x(m + 1) - \frac{2}{231}.$$

$$S_m = \frac{4^m(m + 1)}{\binom{2m}{m}} x(m + 1) - \frac{2}{231}.$$

$$S_m = \frac{2(4^m)(m + 1)(63m^4 + 112m^3 + 18m^2 - 22m + 3)}{\binom{2m}{m}693} - \frac{2}{231}.$$
Example 10. Use Gosper’s algorithm to find a closed form representation of the sum

\[ S_m = \sum_{n=0}^{m} \frac{(2n)^2}{(n+1)4^{2n}}. \]

In Step 1 we form \( r(n) = \frac{\ell_{n+1}}{\ell_n} \) and simplify with algebra. \( r(n) \) equals

\[
\frac{(2n+2)^2}{(n+2)4^{2n}} \frac{(n+1)^2}{16(n+1)^3(n+2)} = \frac{(2n+1)^2}{4(n+1)(n+2)} = \frac{(n+\frac{1}{2})^2}{(n+1)(n+2)}.
\]

Step 2 requires that we write \( r(n) \) in the form of (4.3) above. Now consider \( \gcd(f(n), g(n + h)) \). We would like to know if this \( \gcd \) is equal to 1 for all nonnegative integers \( h \). If so, we would have our simple factorization and move on to step 3.

\[
\gcd((n + \frac{1}{2})^2, (n + h + 1)(n + h + 2)) = 1.
\]

Since the \( \gcd \) is 1 for all positive integers \( h \) we need not follow the extended procedure for step 2. We simply have \( a(n) = f(n) \), \( b(n) = g(n) \), and \( c(n) = 1 \). If it is too difficult to verify that the \( \gcd \) is 1, simply continue with the extended algorithm of step 2. You will come to the same result. We now have \( a(n) = (n + \frac{1}{2})^2 \), \( b(n) = (n + 1)(n + 2) \), and \( c(n) = 1 \).

Now we know the form of the equation that we must use to find \( x(n) \).

\[
(n + \frac{1}{2})^2 x(n + 1) - n(n + 1) x(n) = 1.
\]

Before looking at the directions for step 3, we note the following: \( \deg a(n) = \deg b(n) = 2 \), \( lc a(n) = lc b(n) = 1 \), \( A = 1 \), and \( B = 1 \). Step 3.a. does not hold, so 3.b. does. Thus

\[
D = \left\{ 0 - 2 + 1, \frac{1 - 1}{1} \right\} = \{-1, 0\}.
\]
When we remove the numbers that are not positive integers or zero, what we have left is the integer 0. So the degree of \( x(n) \) is zero — the function will be a constant function. Now use the method of undetermined coefficients and solve the following equation.

\[
(n + \frac{1}{2})^2(c_0) - n(n + 1)(c_0) = 1.
\]

\[
c_0(n^2 + n + \frac{1}{4}) - c_0(n^2 + n) = 1.
\]

\[
x(n) = c_0 = 4.
\]

Step 4 is to return the value of \( z_n = \frac{b(n-1)x(n)}{c(n)} t_n \) which in our case is

\[
z_n = \frac{4n(n+1) \binom{2n}{n}^2}{(n+1)4^{2n}} = \frac{n \binom{2n}{n}^3}{4^{2n-1}}.
\]

Again Gosper's algorithm returns the value of \( z_n \) as we had hoped. Since we want the sum of \( m + 1 \) terms, we use \( S_m = s_{m+1} = z_{m+1} - z_0 \).

\[
S_m = \frac{(m+1) \binom{2m+2}{m+1}^2}{4^{2m+1}} - \frac{0 \binom{0}{0}^2}{4^1}.
\]

\[
S_m = \frac{(m+1) \binom{2m+2}{m+1}^2}{4^{2m+1}}.
\]

\[
S_{m} = \frac{(m+1)(2m+2)^2(2m+1)^2 \binom{2m}{m}^2}{4^{2m+1}(m+1)^4}.
\]

\[
S_{m} = \frac{(2m+1)^2 \binom{2m}{m}^2}{4^{2m}(m+1)}.
\]
4.3 Analysis of Gosper’s Algorithm

Many of the details of Gosper’s algorithm took some work to prove as true. We now examine some of the foundational proofs that allow us to move forward at different steps in the algorithm.

**Theorem 4.1** Let \( a(n) \), \( b(n) \), \( c(n) \) be polynomials in \( n \) such that (4.4) holds. If \( x(n) \) is a rational function of \( n \) satisfying (4.6), then \( x(n) \) is a polynomial in \( n \).

This theorem is needed to guarantee that the solution we find for \( x(n) \) in step 3 is in fact a polynomial solution. A proof by contradiction was given by Gosper in 1978. Let \( x(n) = \frac{f(n)}{g(n)} \). Suppose that \( g(n) \) is a non-constant polynomial. Multiply through (4.6) by \( g(n)g(n+1) \), and consider a \( u(n) \), a non-constant irreducible common divisor of \( g(n) \) and \( g(n+N) \). Using rules of division, \( u(n+1) \) is found to divide both \( a(n) \) and \( b(n+N) \), which is a contradiction of (4.4).

**Proposition 4.2** Let \( 0 \leq k \leq i, j \leq N, h \in \mathbb{Z}^+ \cup \{0\} \) and \( h < h_{k+1} \). Then \( \text{gcd}(p_i(n), q_j(n+h)) = 1 \).

This verifies that the \( p_i \)'s and \( q_i \)'s in step 2 of Gosper’s algorithm have no common factor. Therefore \( a(n) \) and \( b(n) \) (from the end of step 2) will also have no common factor.

**Theorem 4.3** Let \( K \) be a field of characteristic zero and \( r \in K [n] \) a nonzero rational function. Then there exist polynomials \( a, b, c \in K [n] \) such that \( b, c \) are
monic and
\[ r(n) = \frac{a(n) c(n + 1)}{b(n) c(n)}, \]
where

1. \( \gcd(a(n), b(n + h)) = 1 \) for every nonnegative integer \( h \),

2. \( \gcd(a(n), c(n)) = 1 \),

3. \( \gcd(b(n), c(n + 1)) = 1 \).

Such polynomials are constructed by Step 2 of Gosper's algorithm.

1. Prove this by letting \( i = j = N \) in Proposition 4.2.

2. Assume that \( a(n) \) and \( c(n) \) have a nonconstant common factor. Then so do \( p_N(n) \) and \( s_i(n - j) \), and from there, so do \( p_N(n) \) and \( q_{i-1}(n + h_i - j) \), which contradicts Proposition 4.2.

3. Similarly to 2, assuming that \( b(n) \) and \( c(n + 1) \) have a nonconstant common factor will lead to a contradiction of Proposition 4.2.

**Lemma 4.4** Let \( K \) be a field of characteristic zero. Let \( a, b, c, A, B, C, \in K[n] \) be polynomials such that \( \gcd(a(n), c(n)) = \gcd(b(n), c(n + 1)) = \gcd(A(n), B(n + h)) = 1 \), for all nonnegative integers \( h \). If
\[
\frac{a(n) c(n + 1)}{b(n) c(n)} = \frac{A(n) C(n + 1)}{B(n) C(n)},
\]
then \( c(n) \) divides \( C(n) \).

**Corollary 4.5** Let \( r(n) \) be a rational function. Then the factorization \( r(n) = \frac{a(n) c(n + 1)}{b(n) c(n)} \), in Theorem 4.3 is unique.
Corollary 4.6 Among all triples \( a(n), b(n), c(n) \) satisfying \( r(n) = \frac{a(n) c(n+1)}{b(n) c(n)} \), and (4.4), the one constructed in Step 2 of Gosper’s algorithm has \( c(n) \) of least degree.

Gosper’s algorithm does definitively answer the question for which it was developed. It tells us whether a particular hypergeometric term can be indefinitely summed. This problem is somewhat analogous to the problem of solving an indefinite integral. This leads us to wonder about what happens when we want the definite sum of a hypergeometric term (which would be analogous to solving a definite integral). Gosper’s sum doesn’t help us here, but our next algorithm will.
5. Zeilberger’s Algorithm

Zeilberger’s algorithm will help us find the definite sum of a hypergeometric term. It uses Gosper’s algorithm and also expands upon the ideas of Sister Mary Celine Fasenmeyer’s method. Again, this algorithm completely answers the question of finding a definite sum of a hypergeometric term.

5.1 The Algorithm

Consider the sum

\[ f(n) = \sum_k F(n, k). \]

Like Sister Celine’s method, we must first have a doubly hypergeometric term, i.e., the ratio \( \frac{F(n, k+1)}{F(n, k)} \) and the ratio \( \frac{F(n+1, k)}{F(n, k)} \) are both hypergeometric. Zeilberger’s algorithm is also referred to as the Creative Telescoping method as we shall later see. Like Sister Celine’s method, Zeilberger first finds a recurrence for the summand \( F(n, k) \) and then sums over \( k \) to find the appropriate recurrence for the sum \( f(n) \). While we can’t expect to always find a nice simple recurrence like these

\[ F(n, k) = G(n, k + 1) - G(n, k) \]

\[ F(n + 1, k) - F(n, k) = G(n, k + 1) - G(n, k), \]

we are guaranteed by Theorem 3.2 to find some recurrence. In terms of shift operators \( K : g(n, k) \rightarrow g(n, k + 1) \) and \( N : f(n, k) \rightarrow f(n + 1, k) \), and a difference operator \( p : (n, N) \rightarrow a_0(n) + a_1(n)N + a_2(n)N^2 + \ldots + a_r(n)N^r \), the
recurrence we are guaranteed to find is

\[ p(n, N)F(n, k) = (K - 1)G(n, k). \]

The coefficients \( a_i(n) \) are polynomials in \( n \), and \( \frac{G(n, k)}{F(n, k)} \) is a rational function of \( n \) and \( k \), so we have

\[ \sum_{j=0}^{J} a_j(n)F(n + j, k) = G(n, k + 1) - G(n, k). \]  \hspace{1cm} (5.1)

Once we find, through Zeilberger's algorithm, the desired recurrence, we try
to sum over \( k \) to get our desired answer. If by chance our recurrence is a simple
linear recurrence, we are happy as that is easy to solve. If the recurrence is not
linear but has constant coefficients, we are also comfortable. If neither of these
situations hold, we still do not worry as that is where Petkovsek's algorithm
"Hyper" comes into play. This algorithm will finish what Zeilberger started and
give us a complete answer to our question.

Begin with the recurrence (5.1) of order \( J \). Fix \( J \) and look for a recurrence.
(If we find no recurrence of order \( J \) then we next look for a recurrence of order
\( J + 1 \).) For our fixed \( J \), the left side of (5.1) can be expressed as

\[ t_k = a_0F(n, k) + a_1F(n + 1, k) + \ldots + a_JF(n + J, k). \]  \hspace{1cm} (5.2)

Then the consecutive term ratio is creatively written as

\[ \frac{t_{k+1}}{t_k} = \frac{\sum_{j=0}^{J} a_jF(n + j, k + 1) / F(n, k + 1) F(n, k + 1)}{\sum_{j=0}^{J} a_jF(n + j, k) / F(n, k)}. \]  \hspace{1cm} (5.3)

The second fraction on the right is a rational function of \( n \) and \( k \), so write it as

\[ \frac{F(n, k + 1)}{F(n, k)} = r_1(n, k) \]

\[ r_2(n, k). \]
where the $r$'s are polynomials. Since $F(n, k)$ was doubly hypergeometric we also have that

\[
\frac{F(n, k)}{F(n-1, k)} = \frac{s_1(n, k)}{s_2(n, k)},
\]

where the $s$'s are polynomials. Then

\[
\frac{F(n+j, k)}{F(n, k)} = \prod_{i=0}^{j-1} \frac{F(n+j-i, k)}{F(n+i, k)} = \prod_{i=0}^{j-1} \frac{s_1(n+j-i, k)}{s_2(n+j-i, k)}.
\]

With some substitution and simplification, we manipulate the consecutive term ratio into the form

\[
\frac{t_{k+1}}{t_k} = \frac{p_0(k+1)}{p_0(k)} \frac{r(k)}{s(k)},
\]

where the following equations hold.

\[
p_0(k) = \sum_{j=0}^{J} \alpha_j \prod_{i=0}^{j-1} s_1(n+j-i, k) \prod_{i=j}^{J} s_2(n+j-i, k).
\]

\[
r(k) = r_1(n, k) \prod_{i=0}^{J} s_2(n+j-i, k).
\]

\[
s(k) = r_2(n, k) \prod_{i=0}^{J} s_2(n+j-i, k+1).
\]

We know we can write $\frac{r(k)}{s(k)}$ in the canonical form of Theorem 4.3 so that

\[
\frac{r(k)}{s(k)} = \frac{p_1(k+1)}{p_1(k)} \frac{p_2(k)}{p_3(k)}.
\]

Now take $p(k) = p_0(k)p_1(k)$ and from (5.4) and (5.5) we get:

\[
\frac{t_{k+1}}{t_k} = \frac{p(k+1)}{p(k)} \frac{p_2(k)}{p_3(k)},
\]

which is now in a form ready for Step 3 of Gosper's algorithm.
Performing step 3 of Gosper’s algorithm, we find an upper bound, $\Delta$, on the degree of our unknown equation $\ell(k)$ such that

$$p_2(k)\ell(k + 1) - p_3(k - 1)\ell(k) = p(k).$$

What will result is a system of linear equations in $J + \Delta + 2$ variables: $\Delta + 1$ from $\ell(k)$ and $J + 1$ from the coefficients $a_j(n)$. If a solution to this system does not exist, then no telescoped recurrence of order $J$ exists, so we seek a recurrence of order $J + 1$. If however, we are able to solve this system of equations then we have found all the $a_j(n)$ coefficients from (5.1). Then we can find $G(n, k)$ from the following:

$$G(n, k) = \frac{p_3(k - 1)}{p(k)} \ell(k)t_k.$$

Then we will sum both sides of (5.1) over $k$ to get

$$\sum_{j=0}^{J} a_j(n)f(n + j) = 0. \quad (5.6)$$

Hopefully, from this point we can easily rewrite the recurrence to find our sum.

### 5.2 An example of Zeilberger’s Algorithm

To compare this algorithm with Sister Celine’s, we now perform Zeilberger’s algorithm on the same simple sum. Let

$$f(n) = \sum_k \binom{n}{k}.$$

Then fix $J = 1$ and the consecutive term ratio is

$$\frac{t_{k+1}}{t_k} = \frac{a_0\binom{n}{k+1} + a_1\binom{n+1}{k+1}}{a_0\binom{n}{k} + a_1\binom{n+1}{k}}.$$
We can expand and simplify this ratio to
\[
\frac{t_{k+1}}{t_k} = \frac{a_0(n-k) + a_1(n+1)}{a_0(n-k+1) + a_1(n+1)} \frac{n-k+1}{k+1}.
\]
In terms of the general directions, we now have \(p_0(k) = a_0(n-k+1) + a_1(n+1),\)
\(r(k) = n - k + 1,\) and \(s(k) = k + 1.\) When we rewrite \(\frac{r(k)}{s(k)}\) in the canonical form
of Theorem 4.3 we simply get that \(p_1(k) = 1, p_2(k) = n - k + 1, p_3(k) = k + 1,\)
and thus \(p(k) = p_0(k)p_1(k) = p_0(k).\) The equation we must use to solve for \(\ell(k)\)
is
\[
p_2(k)\ell(k+1) - p_3(k-1)\ell(k) = p(k).
\]
\[(n-k+1)\ell(k+1) - k\ell(k) = a_0(n-k+1) + a_1(n+1).
\]
Step 3 of Gosper's algorithm gives us an upper bound of 0 for the degree of \(\ell(k).\)
Thus \(\ell(k) = \text{constant}.\) Let \(\beta\) be that constant. Then
\[
\beta(n-k+1) - \beta k = a_0(n-k+1) + a_1(n+1).
\]
\[k(-2\beta) + (\beta(n+1)) = k(-a_0) + (a_0(n+1) + a_1(n+1)).
\]
This gives the following system of equations.
\[
-2\beta = -a_0 \quad (5.7)
\]
\[
\beta(n+1) = a_0(n+1) + a_1(n+1) \quad (5.8)
\]
\(2\beta = a_0\) and \(\beta = a_0 + a_1\) implies \(\beta = -a_1.\) So, in terms of \(\beta\) we have \(a_0 = 2\beta,\)
\(a_1 = -\beta,\) and \(\beta = \beta.\) Now our recurrence in the form of 5.6 is
\[
2\beta f(n) - \beta f(n+1) = 0.
\]
Hence, \(f(n+1) = 2f(n),\) and using the initial value of \(f(0) = 1,\) we obtain
\[
f(n) = 2^n.
\]
5.3 Analysis of Zeilberger’s algorithm

Zeilberger’s algorithm, like Gosper’s algorithm, completely answers the question at hand. We are guaranteed a recurrence of the form (5.1) by Theorem 5.1 which we will soon examine. Once we have the recurrence relation, we can either find the closed form by hand or if the recurrence is complex we will use Petkovsek’s algorithm “Hyper.” The algorithm “Hyper” will either return the solution of the recurrence, \( f(n) \) (which could be a linear combination of a fixed number of hypergeometric terms), or will state that no such solution exists.

**Theorem 5.1** Let \( F(n, k) \) be a proper hypergeometric term. Then \( F \) satisfies a nontrivial recurrence of the form (5.1), in which \( \frac{G(n, k)}{F(n, k)} \) is a rational function of \( n \) and \( k \).

This theorem is proved using operator form. We have the shift operators \( N \) and \( K \) as above, and so we write (3.7), the two term recurrence from Theorem 3.2, as

\[
P(N, n, K)F(n, k) = 0.
\]

In other terminology, \( P \) is an operator that annihilates \( F(n, k) \). We assume that \( P \) is the annihilator of \( F(n, k) \) with the least degree in \( K \). \( P \) is manipulated using power series expansion so that we have

\[
0 = P(N, n, K)F(n, k) = (P(N, n, 1) + (1 - K)Q(N, n, K))F(n, k);
\]

where \( Q \) is a polynomial. Then

\[
P(N, n, 1)F(n, k) = (K - 1)Q(N, n, K)F(n, k).
\]
The case where \( P(N, n, 1) = 0 \) and the case where \( Q(N, n, K) = 0 \) are both examined and found to possess nonzero \( k \)-free operators which annihilate \( F(n, k) \) with degree less than \( P \). This is a contradiction of our original assumptions about \( P \).
6. WZ Algorithm

The WZ algorithm provides a proof certificate as a way to verify the truth of a combinatorial identity. Differing from the previous two algorithms, the main purpose of WZ is verification of an identity rather than solving a sum to create an identity. As we will see, the WZ algorithm also makes use of Gosper’s algorithm. And after the basic verification, there are a few additional identities that you get for free. What is impressive is that the proof certificate consists of a single rational function, \( R(n, k) \). Thus we have a great way to find a concise proof of a known identity. This method was developed by Herbert Wilf and Doron Zeilbergerer, thus the name “WZ.”

6.1 The Algorithm

What we want to prove is that

\[
\sum_k f(n, k) = r(n).
\]

If \( r(n) \neq 0 \), then divide both sides through by \( r(n) \). Now think of the left hand side as the new summand and we have

\[
\sum_k \frac{f(n, k)}{r(n)} = \sum_k F(n, k) = 1.
\]

So now what we want to prove is that \( f(n) = \sum_k F(n, k) \) is constant which we will do by showing \( f(n + 1) - f(n) = 0 \) for all \( n \). If we can find a term \( G(n, k) \) such that

\[
F(n + 1, k) - F(n, k) = G(n, k + 1) - G(n, k), \tag{6.1}
\]
then when we sum both sides of the equation over \( k \), we get \( f(n+1) - f(n) = 0 \).
A pair of functions, \( F \) and \( G \), which satisfy (6.1) is called a WZ Pair. The third function of relevance for this discussion is \( R(n, k) = \frac{G(n, k)}{F(n, k)} \), the proof certificate.

First we mention how to verify an identity from its proof certificate and second we will consider how to form the certificate, \( R(n, k) \).

To prove an identity \( \sum_k f(n, k) = r(n) \) from its proof certificate \( R(n, k) \), do the following.

1. If \( r(n) \neq 0 \), then put \( F(n, k) = \frac{f(n, k)}{r(n)} \), else put \( F(n, k) = f(n, k) \). Let \( G(n, k) = R(n, k)F(n, k) \).

2. Verify that (6.1) is true. Write everything out, simplify, and verify the polynomial identity.

3. Verify that the given identity is true for one value of \( n \).[7]

These steps are straightforward and only require algebra. As the expressions can be quite complex it is advisable to use a math software program.

To find a proof certificate for an identity do the following.

1. If \( r(n) \neq 0 \), then put \( F(n, k) = \frac{f(n, k)}{r(n)} \), else put \( F(n, k) = f(n, k) \).

2. Let \( f(k) = F(n+1, k) - F(n, k) \). Input \( f(k) \) into Gosper’s algorithm. If Gosper’s algorithm fails, then the WZ algorithm will too.

3. If successful, the output \( G(n, k) \) of Gosper’s algorithm is the WZ mate of \( F \). The rational function \( R(n, k) = \frac{G(n, k)}{F(n, k)} \) is the WZ certificate of the identity \( \sum_k F(n, k) = constant \)[7].
6.2 An Example of the WZ Algorithm

Example 11. Suppose you are given the identity

\[ \sum_k \binom{n}{k}^2 = \binom{2n}{n}, \]

and the proof certificate

\[ R(n, k) = -\frac{k^2(3n - 2k + 3)}{2(2n + 1)(n - k + 1)^2}. \]

Then in order to verify this identity follow the appropriate steps above. First write \( F(n, k) = \binom{n}{k}, \) and \( G(n, k) = R(n, k)F(n, k) \) and simplify.

\[ F(n, k) = \frac{n!}{(2n)!} \frac{n!}{(n-k)!}, \quad \frac{n!}{(n-k)!} = \frac{(n-k)!}{(2n)!} \]

\[ G(n, k) = R(n, k)F(n, k) = \left( -\frac{k^2(3n - 2k + 3)}{2(2n + 1)(n - k + 1)^2} \right) \left( \frac{n!}{(2n)!} \frac{n!}{(n-k)!} \right) \]

\[ = -\frac{k^2(3n - 2k + 3)n!}{2(2n + 1)(n - k + 1)^2k!}. \]

Now substitute into (6.1), simplify, and verify the polynomial identity.

\[ \frac{(n+1)!}{(2n+2)!k!(n+1-k)!} = \frac{n!}{(2n)!k!(n-k)!} \]

\[ = -\frac{(3n - 2k + 1)n!}{2(2n + 1)(n - k)!k!^2} + \frac{k^2(3n - 2k + 3)n!}{2(2n + 1)!k!(n - k + 1)!^2}. \]

Now multiply both sides by \( \frac{k^2(n-k+1)!^2(2n+1)!}{n!}. \) to get

\[ (n+1)^4 - (n - k + 1)^2(2n+2)(2n+1) \]

\[ = -(3n - 2k + 1)(n+1)(n - k + 1)^2 + k^2(3n - 2k + 3)(n+1). \]

Divide both sides by \( n+1. \)

\[ (n+1)^3 - 2(n - k + 1)^2(2n+1) = -(3n - 2k + 1)(n - k + 1)^2 + k^2(3n - 2k + 3). \]
Using Derive 6,

$$k^2(-4n - 2) + k(8n^2 + 12n + 4) - 3n^3 - 7n^2 - 5n - 1$$

$$= k^2(-4n - 2) + k(8n^2 + 12n + 4) - 3n^3 - 7n^2 - 5n - 1,$$

so the polynomial identity holds. Now we must check the identity for one value of $n$. Let $n = 3$. Then we have

$$\sum_k \binom{3}{k}^2 = \binom{3}{0}^2 + \binom{3}{1}^2 + \binom{3}{2}^2 + \binom{3}{3}^2 = 1^2 + 3^2 + 3^2 + 1^2 = 20,$$

and

$$\binom{6}{3} = \frac{6!}{3!3!} = \frac{6 \cdot 5 \cdot 4}{3 \cdot 2 \cdot 1} = 20,$$

so we are finished and have completely verified the given identity.

### 6.3 Analysis of the WZ Algorithm

One issue with the WZ algorithm is that it depends upon Gosper's algorithm. As we saw in the steps to finding $R(n, k)$, if Gosper's algorithm fails, then so will this one. Now, we always have a recurrence of the form

$$\sum_{j=0}^f a_j(n)F(n + j, k) = G(n, k + 1) - G(n, k),$$

and it has been observed [7] that 99% of the time the left hand side will reduce to two terms to give us the WZ equation(6.1).

The steps for verifying an identity and also for finding a proof certificate are based on the following theorem. But first, we list some items referenced in the theorem.

1. For each integer $k$, the limit

$$f_k = \lim_{n \to \infty} F(n, k)$$

exists and is finite.
2. For each integer \( n \geq 0 \), we have

\[
\lim_{k \to \pm \infty} G(n, k) = 0.
\]

3. We have \( \lim_{L \to \infty} \sum_{n \geq 0} G(n, -L) = 0 \).

**Theorem 6.1** Let \((F, G)\) satisfy (6.1). If 2 above holds, then

\[
\sum_{k} F(n, k) = \text{constant} \quad (n = 0, 1, 2, \ldots).
\]  

(6.2)

If 1 and 3 above both hold, then we have the companion identity

\[
\sum_{n \geq 0} G(n, k) = \sum_{j \leq k-1} (f_j - F(0, j)),
\]  

(6.3)

where \( f \) is defined in 1 above.

This companion identity is one of the “free” identities we get when using the WZ algorithm. This identity comes about because of the symmetry \( n \) and \( k \) have in (6.1).

**Proof:** We use the symbol \( \Delta_n \) for the forward difference operator on \( n \):

\( \Delta_n h(n) = h(n + 1) - h(n) \). Sum both sides of the WZ equation from \( k = -L \) to \( k = K \), getting

\[
\Delta_n \left\{ \sum_{k=-L}^{K} F(n, k) \right\} = \sum_{k=-L}^{K} \left\{ \Delta_k G(n, k) \right\} = G(n, K + 1) - G(n, -L).
\]

Now let \( K, L \to \infty \) and use 2 from above to find that \( \Delta_n \sum_k F(n, k) = 0 \), i.e., that \( \sum_k F(n, k) \) is independent of \( n \geq 0 \), which establishes (6.2). If we sum both sides of the WZ equation from \( n = 0 \) to \( N \), we get

\[
F(N + 1, k) - F(0, k) = \Delta_k \left\{ \sum_{n \geq 0} G(n, k) \right\}.
\]
Now let $N \to \infty$ and use 1 above to get

$$f_k - F(0, k) = \Delta_k \left\{ \sum_{n \geq 0} G(n, k) \right\}.$$ 

Replace $k$ by $k'$, sum from $k' = -L$ to $k' = k - 1$, let $L \to \infty$, and use 3 above to obtain (6.3). This proof is taken from [7].

As an illustration of the companion identity, consider our previous identity

$$\sum_k \binom{n}{k}^2 = \binom{2n}{n}.$$ 

Recall that $F(n, k) = \binom{n}{k}^2 \binom{2n}{n}$, and note that (using 1 above) $f_k = 0$ for all values of $k$. $G(n, k)$ can be written as

$$G(n, k) = \frac{(-3n + 2k - 3)}{2(2n + 1)} \frac{n!^2}{(k - 1)!^2(n - k + 1)!^2 \binom{2n}{n}}.$$ 

The companion identity we seek is of the form

$$\sum_{n \geq 0} G(n, k) = \sum_{j \leq k - 1} (f_j - F(0, j)).$$

So we have

$$\sum_{n \geq 0} \frac{(-3n + 2k - 3)}{2(2n + 1)} \frac{n!^2}{(k - 1)!^2(n - k + 1)!^2 \binom{2n}{n}} = \sum_{j \leq k - 1} (0 - F(0, j)) = \begin{cases} 0 & \text{if } k = 0 \\
-1 & \text{if } k \geq 1, \end{cases}$$

which simplifies to

$$\sum_{n \geq 0} \frac{(3n - 2k + 1)}{(2n + 1)} \frac{\binom{n}{k}^2}{\binom{2n}{n}} = 2.$$ 

So the above identity is new and we get it free when we use the WZ algorithm.

Another identity that we will not explore here, is the dual identity of a given
identity. Basically, you perform some operations on $F$ and $G$, both functions of a WZ pair, to obtain $F^*$ and $G^*$. The exciting result is that $F^*$ and $G^*$ are also a WZ pair, thus introducing another identity. In [7], several other methods, tricks and ideas for determining more identities are discussed. This would be interesting material on its own for another paper.

In summary, the WZ algorithm helps you to both verify an identity quite easily and determine new identities from your original.
7. Algorithm “Hyper”

This algorithm is used after Zeilberger’s algorithm when the recurrence returned falls into one of the more complex categories.

Through a strictly computational approach, algorithm Hyper finds a hypergeometric solution of $Ly = f$ where $L$ is a linear recurrence operator. For additional clarification, consider that

$$L = \sum_{i=0}^{r} p_i(n) N^i$$

where $N$ is the shift operator mentioned in the explanation of Zeilberger’s algorithm.

Since the algorithm Hyper makes use of a subroutine, algorithm Poly, we consider the subroutine first.

7.1 “Poly”

The input is the set of polynomials, $p_i(n)$ over $F$, a field, for $i = 0, 1, \ldots, r$, where $r$ is the degree of $L$. The output is the general polynomial solution of $Ly = f$. To perform this algorithm, use the following steps.

1. Compute $q_j = \sum_{i=j}^{r} \binom{i}{j} p_i$, for $0 \leq j \leq r$.

2. Let $\text{deg} \ 0 = -\infty$ and let $b = \max_{0 \leq j \leq r} \{ \text{deg} \ q_j - j \}$.

3. Let \( \alpha(x) = \sum_{\substack{0 \leq j \leq r \\text{deg} \ q_j - j = b}} \text{lc} \ (q_j) \ x^j \), where $\text{lc} \ (q_j)$ denotes the leading coefficient of the polynomial $q_j$, and $x^j$ is the falling factorial $x(x-1)\ldots(x-j+1)$. 

47
4. Let \( d_1 = \max \{ x \in \mathbb{Z}^+ \cup \{ 0 \} : \alpha(x) = 0 \} \).

5. Compute \( d \) using \( d = \max \{ \deg f - b, -b - 1, d_1 \} \).

6. Using the method of undetermined coefficients, find all \( y(n) \) of the form
\[
y(n) = \sum_{k=0}^{d} c_k n^k
\]
that satisfy \( Ly = f \).

**Example 12.** Find the polynomial solutions of
\[
3y(n + 2) - ny(n + 1) + (n - 1)y(n) = 0.
\]

Use the steps above to identify the following.

\( p_0(n) = n - 1 \), \( p_1(n) = -n \), and \( p_2(n) = 3 \).

\[
q_0 = \sum_{i=0}^{2} \binom{i}{0} p_i = \binom{0}{0} p_0 + \binom{1}{0} p_1 + \binom{2}{0} p_2 = p_0 + p_1 + p_2 = n - 1 - n + 3 = 2.
\]

\[
q_1 = \sum_{i=1}^{2} \binom{i}{1} p_i = \binom{1}{1} p_1 + \binom{2}{1} p_2 = p_1 + 2p_2 = -n + 2(3) = 6 - n.
\]

\[
q_2 = \sum_{i=2}^{2} \binom{i}{2} p_i = \binom{2}{2} p_2 = p_2 = 3.
\]

\( \deg 0 = -\infty \), and \( b = \max \{ 0 - 0, 1 - 1, 0 - 2 \} = 0 \).

\( \alpha(x) = 2x^0 - x^1 = 2 - x \), and \( d_1 = \max \{ 2 \} = 2 \).

\( d = \max \{ -\infty - 0, -0 - 1, 2 \} = 2 \).

So now we can use the method of undetermined coefficients to find
\[
y(n) = c_0 n^0 + c_1 n^1 + c_2 n^2
\]
satisfying
\[
3(c_0 + c_1(n + 2)^1 + c_2(n + 2)^2) - n(c_0 + c_1(n + 1)^1 + c_2(n + 1)^2) + c_2(n + 1)^2 + (n - 1)(c_0 + c_1 n^1 + c_2 n^2) = 0.
\]
Equating the coefficients of powers of \( n \) to zero yields the following system of equations:

\[
2c_0 + 6c_1 + 12c_2 = 0 \tag{7.1}
\]
\[
c_1 + 11c_2 = 0. \tag{7.2}
\]

Letting \( c_2 = 1 \) gives

\[
y(n) = 27 - 11n + n^2.
\]

7.2 “Hyper”

As previously mentioned, Hyper makes use of Poly as a sort of subroutine. Again, there is much notation that must be organized and prepared in order to clearly navigate the algorithm.

The input is again the set of polynomials, \( p_i(n) \) over \( F \), a field, for \( i = 0, 1, \ldots, d \) where \( d \) is the degree of \( Lg \). Also, input \( K \), an extension field of \( F \). The output this time will be a hypergeometric solution of \( Ly = 0 \) over \( K \) if one exists and 0 otherwise. The execution of this algorithm consists of the following steps.

1. For all monic factors \( a(n) \) of \( p_0(n) \) and \( b(n) \) of \( p_d(n - d + 1) \) over \( K \) do:

   Let \( P_i(n) = p_i(n) \prod_{j=0}^{i-1} a(n + j) \prod_{j=1}^{d-1} b(n + j) \), for \( i = 0, 1, \ldots, d \);

   Let \( m = \max_{0 \leq i \leq d} \deg P_i(n) \);

   Let \( \alpha_i \) be the coefficient of \( n^m \) in \( P_i(n) \), for \( i = 0, 1, \ldots, d \);

   for all nonzero \( z \in K \) such that

   \[
   \sum_{i=0}^{d} \alpha_i z^i = 0 \tag{7.3}
   \]
do:

If the recurrence

\[ \sum_{i=0}^{d} z^i P_i(n)c(n+i) = 0 \]

has a nonzero polynomial solution \( c(n) \) over \( K \) then

let \( S(n) = z(a(n)/b(n))(c(n+1)/c(n)) \);

return a nonzero solution \( y(n) \) of \( y(n+1) = S(n)y(n) \) and

stop.

2. Return 0 and stop.

7.3 Examples Using Algorithms Poly and Hyper

We now attempt an in-depth example using both algorithms.

Example 13.

Find the hypergeometric solution of

\[ (n - 1)y(n + 2) - (n^2 + 3n - 2)y(n + 1) + 2n(n + 1)y(n) = 0. \]

First identify the \( p_i(n) \)'s. \( p_0(n) = 2n(n + 1), \ p_1(n) = -(n^2 + 3n - 2), \)
\( p_2(n) = n - 1. \) Notice that \( d = 2. \) Also list the possible factors \( a(n) \) and \( b(n). \)
\( a(n) \) could equal 1, \( n, n + 1, n(n + 1). \)
\( b(n) \) could equal 1, \( n - 2. \)

This would require that we perform the main loop in Hyper 8 times — once
for each combination of \( a(n) \) and \( b(n). \)

We will perform 2 cases completely and then mention what happens in the
other 6 cases.
Case 1

Let $a(n) = 1$ and $b(n) = 1$. Then $P_0(n) = 2n(n+1)$, $P_1(n) = -(n^2 + 3n - 2)$, and $P_2(n) = n - 1$. Also, $m = 2$, $\alpha_0 = 2$, $\alpha_1 = -1$, and $\alpha_2 = 0$. Thus equation (7.3) is $2 - z = 0$ so $z = 2$.

The recurrence for which we now desire to find polynomial solutions is

$$2n(n+1)c(n) - 2(n^2 + 3n - 2)c(n+1) + 4(n-1)c(n+2) = 0,$$

which reduces to

$$n(n+1)c(n) - (n^2 + 3n - 2)c(n+1) + 2(n-1)c(n) = 0.$$ 

This is the recurrence we will feed into the algorithm Poly.

$p_0(n) = n^2 + n$, $p_1(n) = -(n^2 + 3n - 2)$, and $p_2(n) = 2n - 2$.

$q_0 = p_0 + p_1 + p_2 = 0$, $q_1 = p_1 + 2p_2 = -n^2 + n - 2$, and $q_2 = p_2 = 2n - 2$.

$\deg q = -\infty$, and $b = \max\{0 - 0, 2 -1, 1 -2\} = 1$.

$\alpha(x) = -x^1 = -x$, and $d = \max\{0\} = 0$.

$d = \max\{-\infty - 1, -1 - 1, 0\} = 0$.

So now we can use the method of undetermined coefficients to find

$$y(n) = c_0 n^0$$

satisfying

$$n(n+1)c_0 - (n^2 + 3n - 2)c_0 + 2(n-1)c_0 = 0.$$ 

This statement is valid for all $c_0$, so let $c_0 = 1$. Poly returns the solution $c(n) = 1$.

Now we go back to Hyper to continue. We find that

$$S(n) = \frac{a(n) c(n+1)}{b(n) c(n)} = \frac{2 \frac{1}{1} \frac{1}{1}}{1} = 2.$$ 

Then we have $y(n+1) = 2y(n)$ and so our final solution to case 1 is $y(n) = 2^n$.  

51
Case 2

Let \( a(n) = n \) and \( b(n) = 1 \). Then \( P_0(n) = 2n(n+1) \), \( P_1(n) = -n(n^2+3n-2) \), and \( P_2(n) = (n-1)(n+1) \). Also, \( m = 3 \), \( \alpha_0 = 0 \), \( \alpha_1 = -1 \), and \( \alpha_2 = 1 \). Thus equation (7.3) gives \(-z + z^2 = 0\) so \( z = 1 \).

The recurrence for which we now desire to find polynomial solutions is
\[
2n(n + 1)c(n) - n(n^2 + 3n - 2)c(n + 1) + (n - 1)n(n + 1)c(n + 2) = 0.
\]

This is the recurrence we will feed into the algorithm Poly.

\[
p_0(n) = 2n^2 + 2n, \quad p_1(n) = -(n^3 + 3n^2 - 2n), \quad \text{and} \quad p_2(n) = n^3 - n.
\]

\[
q_0 = p_0 + p_1 + p_2 = -n^2 + 3n, \quad q_1 = p_1 + 2p_2 = n^3 - 3n^2, \quad \text{and} \quad q_2 = p_2 = n^3 - n.
\]

\[
\deg 0 = -\infty, \quad \text{and} \quad b = \max\{2 - 0, \ 3 - 1, \ 3 - 2\} = 2.
\]

\[
\alpha(x) = -x^0 + x^1 = -1 + x, \quad \text{and} \quad d_1 = \max\{1\} = 1.
\]

\[
d = \max\{-\infty - 2, \ -2 - 1, \ 1\} = 1.
\]

So now we can use the method of undetermined coefficients to find
\[
y(n) = c_0n^0 + c_1n^1
\]
satisfying
\[
2n(n + 1)(c_0 + c_1n^1) - n(n^2 + 3n - 2)(c_0 + c_1(n + 1)^1) + (n - 1)n(n + 1)(c_0 + c_1(n + 2)^1) = 0.
\]

This statement is valid for all \( c_1 \) when \( c_0 = 0 \), so let \( c_1 = 1 \). Poly returns the solution \( c(n) = n \). Now we go back to Hyper to continue. We find that
\[
S(n) = \frac{a(n)c(n + 1)}{b(n)c(n)} = 1 \cdot \frac{n + 1}{n} = \frac{n(n + 1)}{n} = n + 1.
\]

Then we have \( y(n + 1) = (n + 1)y(n) \) and so our final solution to case 2 is \( y(n) = n! \).

Now we examine what happens in the rest of the cases.
<table>
<thead>
<tr>
<th>a(n)</th>
<th>b(n)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n-2</td>
<td>Equation (7.3) has no solution</td>
</tr>
<tr>
<td>n</td>
<td>n-2</td>
<td>z=2, but α(x) = -2 - x which has no nonnegative solutions.</td>
</tr>
<tr>
<td>n+1</td>
<td>1</td>
<td>The algorithm works and we get a previous answer y(n)=n!.</td>
</tr>
<tr>
<td>n+1</td>
<td>n-2</td>
<td>z=2, but α(x) = -3 - x which has no nonnegative solutions.</td>
</tr>
<tr>
<td>n(n+1)</td>
<td>1</td>
<td>Equation (7.3) has no solution</td>
</tr>
<tr>
<td>n(n+1)</td>
<td>n-2</td>
<td>z=1, but α(x) = 2 + x which has no nonnegative solutions.</td>
</tr>
</tbody>
</table>

After inspecting each of the eight cases, we see that the two viable solutions are \( y(n) = n! \) and \( y(n) = 2^n \). Thus, the hypergeometric solution to our original recurrence is \( y(n) = C'2^n + Dn! \), where \( C, D \) are arbitrary constants.
8. Summary

In summary, Gosper's algorithm and Zeilberger's algorithm — together with algorithm "Hyper" — completely answer the questions of summing indefinite and definite hypergeometric sums respectively. The solutions are fully algorithmic and will either return the desired results or return a message saying there is no solution. The inputs must be proper hypergeometric sums and the closed form output will be a linear combination of hypergeometric terms. We do not have a guarantee that a certain sum will be able to be written in hypergeometric closed form, but we do have a theorem guaranteeing we can find a recurrence relation for each sum. The problem sometimes comes when we try to find a solution for the recurrence.

The WZ algorithm significantly cuts down on the length of proofs and gives a proof certificate for which it is easy to verify the proof. It also provides a means to find additional identities — both as a byproduct of the standard operation and as a separate topic on its own.

All in all, even with the few shortcomings they have, these computerized algorithms have tremendously aided mathematicians in their quest to find combinatorial identities.
REFERENCES


