Polynomial Computer Algebra and implmentation of Wilf-Zeilberger's method

Lars Åström

Supervisor: Victor Ufnarovski

Faculty of Engineering at Lund University

lars96astrom@gmail.com

December 19, 2019

Overview

What is the thesis about?

Polynomial Computer Algebra and implementation of Wilf-Zeilberger's method

What is the thesis about?

Polynomial Computer Algebra and implementation of Wilf-Zeilberger's method

What and Why?

WHAT

Show that

$$\sum_{k=n}^{\infty} \frac{1}{\binom{k}{n}} = \frac{n}{n-1}.$$

Characteristics

- Summation on one side.
- Show that...
- Often binomial coefficients.

WHY

- Wilf-Zeilberger's method $\rightarrow$ not so much
- Automized proof generation $\rightarrow$ a lot
- Computer Algebra $\rightarrow$ a lot

What and Why?

WHAT

Show that

$$\sum_{k=n}^{\infty} \frac{1}{\binom{k}{n}} = \frac{n}{n-1}.$$

Characteristics

- Summation on one side.
- Show that...
- Often binomial coefficients.

WHY

- Wilf-Zeilberger's method $\rightarrow$ not so much
- Automized proof generation $\rightarrow$ a lot
- Computer Algebra $\rightarrow$ a lot

What and Why?

WHAT

Show that

$$\sum_{k=n}^{\infty} \frac{1}{\binom{k}{n}} = \frac{n}{n-1}.$$

Characteristics

- Summation on one side.
- Show that...
- Often binomial coefficients.

WHY

- Wilf-Zeilberger's method $\rightarrow$ not so much
- Automized proof generation $\rightarrow$ a lot
- Computer Algebra $\rightarrow$ a lot

What and Why?

WHAT

Show that

$$\sum_{k=n}^{\infty} \frac{1}{\binom{k}{n}} = \frac{n}{n-1}.$$

Characteristics

- Summation on one side.
- Show that...
- Often binomial coefficients.

WHY

- Wilf-Zeilberger's method $\rightarrow$ not so much
- Automized proof generation $\rightarrow$ a lot
- Computer Algebra $\rightarrow$ a lot

What and Why?

WHAT

Show that

$$\sum_{k=n}^{\infty} \frac{1}{\binom{k}{n}} = \frac{n}{n-1}.$$

Characteristics

- Summation on one side.
- Show that...
- Often binomial coefficients.

WHY

- Wilf-Zeilberger's method $\rightarrow$ not so much
- Automized proof generation $\rightarrow$ a lot
- Computer Algebra $\rightarrow$ a lot

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

Important findings

- 1960s: Computer Algebra
- 1978: Gosper's Algorithm
- 1990: Wilf-Zeilberger's method
- 1994: WZ implemented in Mathematica

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

- Used for implementation of WZ
- Polynomial
  $$p(k) = a_0 + a_1 k + \ldots + a_m k^m$$
  is stored as
  $$[a_0, a_1, \ldots, a_m]$$
- Coefficients a_i can be integers or polynomials

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

- Used to prove identities on the form
$$\sum_k F(n,k) = 1$$
- Does this by proving
$$\sum_k F(n+1,k) = \sum_k F(n,k)$$
- Which is done by "changing variables"

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

- An algorithm to find a function S such that
$$a_k = S_k - S_{k-1}$$
- Finds the change of variables needed in WZ

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

- The program writes formal proofs
- Proves 80% of the examples
- The remaining seem impossible to prove by WZ method

Short version of the thesis

- Historical background
- Polynomials
- Wilf-Zeilberger's method
- Gosper's algorithm
- Results
- Conclusions

- The program seems to work well, although cannot solve all examples
- Computer Algebra quickly gets complicated

What problems can be solved?

Problems on the form

$$\sum_k F(n,k) = 1$$

can be solved. Problems on the form

$$\sum_k A(n,k) = B(n)$$

can get converted to the right form.

What problems can be solved?

Problems on the form

$$\sum_k F(n,k) = 1$$

can be solved. Problems on the form

$$\sum_k A(n,k) = B(n)$$

can get converted to the right form.

The idea

Want to prove

$$\sum_k F(n,k) = 1$$

Find $G(n,k)$ such that $F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k)$ and $\lim_{k\to\pm\infty} G(n,k) = 0$.Now

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0.$$

Therefore

$$\sum_k F(n,k)$$

is constant for all n, and if we can evaluate for one n then we are done.

The idea

Want to prove

$$\sum_k F(n,k) = 1$$

Find $G(n,k)$ such that $F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k)$ and $\lim_{k\to\pm\infty} G(n,k) = 0$. Now

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0.$$

Therefore

$$\sum_k F(n,k)$$

is constant for all n, and if we can evaluate for one n then we are done.

The idea

Want to prove

$$\sum_k F(n,k) = 1$$

Find $G(n,k)$ such that $F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k)$ and $\lim_{k\to\pm\infty} G(n,k) = 0$.Now

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0.$$

Therefore

$$\sum_k F(n,k)$$

is constant for all n, and if we can evaluate for one n then we are done.

The idea

Want to prove

$$\sum_k F(n,k) = 1$$

Find $G(n,k)$ such that $F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k)$ and $\lim_{k\to\pm\infty} G(n,k) = 0$.Now

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0.$$

Therefore

$$\sum_k F(n,k)$$

is constant for all n, and if we can evaluate for one n then we are done.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n=0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

Steps of the method

1. Start with
$$\sum_k A(n,k) = B(n)$$
2. Let $F(n,k) = \frac{A(n,k)}{B(n)}$
3. Find $G(n,k)$ such that the conditions are satisfied
4. Show that $\sum_k F(n',k) = 1$ for some n'

1. Show that
$$\sum_k \binom{n}{k} = 2^n$$
2. $F(n,k) = \frac{\binom{n}{k}}{2^n}$
3. Let $G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}}$.
4. For $n = 0$ we have $\sum_k F(n,k) = \frac{\binom{0}{0}}{2^0} = 1$, thus we have proved the identity.

What problems can be solved?

Given an expression a_k, Gosper's algorithm finds an expression S_k such that

$$a_k = S_k - S_{k-1}.$$

With $a_k = F(n+1,k) - F(n,k)$ we get that $G(n,k) = S_{k-1}$ makes the first condition in Wilf-Zeilberger's method fulfilled.

What problems can be solved?

Given an expression a_k, Gosper's algorithm finds an expression S_k such that

$$a_k = S_k - S_{k-1}.$$

With $a_k = F(n+1,k) - F(n,k)$ we get that $G(n,k) = S_{k-1}$ makes the first condition in Wilf-Zeilberger's method fulfilled.

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and $$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$$
2. Find polynomial f_k such that $p_k = q_{k+1} f_k - r_k f_{k-1}$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and $\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$
2. Find polynomial f_k such that $p_k = q_{k+1} f_k - r_k f_{k-1}$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and
$$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$$
2. Find polynomial f_k such that
$$p_k = q_{k+1} f_k - r_k f_{k-1}$$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and $\dfrac{a_k}{a_{k-1}} = \dfrac{p_k}{p_{k-1}} \dfrac{q_k}{r_k}$
2. Find polynomial f_k such that $p_k = q_{k+1} f_k - r_k f_{k-1}$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

Now we see that

$$S_k - S_{k-1} = \frac{q_{k+1}}{p_k} f_k a_k - \frac{q_k}{p_{k-1}} f_{k-1} a_{k-1} =$$

$$= \frac{a_k}{p_k}\left(q_{k+1} f_k - \frac{q_k}{p_{k-1}} f_{k-1} p_k \frac{a_{k-1}}{a_k}\right) =$$

$$= \frac{a_k}{p_k}\left(q_{k+1} f_k - \frac{q_k}{p_{k-1}} f_{k-1} p_k \frac{p_{k-1}}{p_k} \frac{r_k}{q_k}\right) =$$

$$= \frac{a_k}{p_k}\left(q_{k+1} f_k - r_k f_{k-1}\right) = \frac{a_k}{p_k} p_k = a_k,$$

which means that this S_k indeed is a solution.

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and
$$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$$
2. Find polynomial f_k such that
$$p_k = q_{k+1} f_k - r_k f_{k-1}$$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

1. For $\sum_k \binom{n}{k} = 2^n$ we get
$$\frac{a_k}{a_{k-1}} = \frac{(2k-n-1)(n+2-k)}{k(2k-n-3)}$$
which gives us $p_k = 2k - n - 1$, $q_k = n + 2 - k$, $r_k = k$.
2. In
$$2k - n - 1 = (n + 1 - k) f_k - k f_{k-1}$$
we see that $f_k = -1$ gives a solution.
3. Now we get $S_k = -\frac{n+1-k}{2k-n-1} a_k = -\frac{\binom{n}{k}}{2^{n+1}}$, which corresponds to the $G(n, k)$ we got in the previous example.

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and
$$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$$
2. Find polynomial f_k such that
$$p_k = q_{k+1} f_k - r_k f_{k-1}$$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

1. For $\sum_k \binom{n}{k} = 2^n$ we get
$$\frac{a_k}{a_{k-1}} = \frac{(2k-n-1)(n+2-k)}{k(2k-n-3)}$$
which gives us $p_k = 2k - n - 1$, $q_k = n + 2 - k$, $r_k = k$.
2. In
$$2k - n - 1 = (n + 1 - k) f_k - k f_{k-1}$$
we see that $f_k = -1$ gives a solution.
3. Now we get $S_k = -\frac{n+1-k}{2k-n-1} a_k = -\frac{\binom{n}{k}}{2^{n+1}}$, which corresponds to the $G(n,k)$ we got in the previous example.

Steps of the algorithm

1. Find polynomials p_k, q_k, r_k such that $gcd(q_k, r_{k+j}) = 1$ $\forall\, j \geq 0$ and
$$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k}$$
2. Find polynomial f_k such that
$$p_k = q_{k+1} f_k - r_k f_{k-1}$$
3. Let $S_k = \frac{q_{k+1}}{p_k} f_k a_k$

1. For $\sum_k \binom{n}{k} = 2^n$ we get
$$\frac{a_k}{a_{k-1}} = \frac{(2k-n-1)(n+2-k)}{k(2k-n-3)}$$
which gives us $p_k = 2k - n - 1$, $q_k = n + 2 - k$, $r_k = k$.
2. In
$$2k - n - 1 = (n + 1 - k) f_k - k f_{k-1}$$
we see that $f_k = -1$ gives a solution.
3. Now we get $S_k = -\frac{n+1-k}{2k-n-1} a_k = -\frac{\binom{n}{k}}{2^{n+1}}$, which corresponds to the $G(n, k)$ we got in the previous example.

2300 lines of code

- 50% methods for polynomials and Wilf-Zeilberger's method
- 20% parsing
- 30% testing of the methods

2300 lines of code

- 50% methods for polynomials and Wilf-Zeilberger's method
- 20% parsing
- 30% testing of the methods

2300 lines of code

- 50% methods for polynomials and Wilf-Zeilberger's method
- 20% parsing
- 30% testing of the methods

2300 lines of code

- 50% methods for polynomials and Wilf-Zeilberger's method
- 20% parsing
- 30% testing of the methods

Results as statistics

- 10 examples for training, 10 for validation
- The automatic solver manages to prove 8 of each
- The remaining examples seem to be unsolvable using Wilf-Zeilberger's method

Results as statistics

- 10 examples for training, 10 for validation
- The automatic solver manages to prove 8 of each
- The remaining examples seem to be unsolvable using Wilf-Zeilberger's method

Results as statistics

- 10 examples for training, 10 for validation
- The automatic solver manages to prove 8 of each
- The remaining examples seem to be unsolvable using Wilf-Zeilberger's method

Results as an example

$k < 1$ and $k > n+1$ gives that

$$\binom{n}{k-1} = 0, \implies \lim_{k \to \pm\infty} G(n,k) = 0.$$

Proof

Automatic WZ-method prover

2019-11-25

We want to prove that

$$\sum (-1)^k \cdot \binom{n}{k}\binom{2k}{k} 4^{n-k} = \binom{2n}{n} \tag{1}$$

holds. By dividing equation 1 by the right hand side we get

$$F(n,k) = \frac{(-1)^k \cdot \binom{n}{k}\binom{2k}{k} 4^{n-k}}{\binom{2n}{n}} \tag{2}$$

We use proof certificate

$$R(n,k) = \frac{2k-1}{2n+1}, \tag{3}$$

which is the same as using

$$G(n,k) = \frac{2k-1}{2n+1} \frac{(-1)^{k-1} \cdot \binom{n}{k-1}\binom{2(k-1)}{k-1} 4^{n-(k-1)}}{\binom{2n}{n}}, \tag{4}$$

the automatic solver has verified that

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k). \tag{5}$$

Thereafter user now has to verify that

$$\lim_{k \to \pm\infty} G(n,k) = 0 \ \ \forall\, n. \tag{6}$$

Then we get

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0 \tag{7}$$

Lastly equation 1 needs to be verified for some n, for instance $n = 0$. Thereafter the identity is shown.

For $n = 0$ we get

$$\sum_k (-1)^k \binom{n}{k}\binom{2k}{k} 4^{n-k} = (-1)^0 \binom{0}{0}\binom{0}{0} 4^0 = 1$$

Also $\binom{0}{0} = 1$.

Results as an example

$k < 1$ and $k > n+1$ gives that

$$\binom{n}{k-1} = 0, \implies \lim_{k\to\pm\infty} G(n,k) = 0.$$

Proof

Automatic WZ-method prover

2019-11-25

We want to prove that

$$\sum (-1)^k \cdot \binom{n}{k}\binom{2k}{k} 4^{n-k} = \binom{2n}{n} \tag{1}$$

holds. By dividing equation 1 by the right hand side we get

$$F(n,k) = \frac{(-1)^k \cdot \binom{n}{k}\binom{2k}{k} 4^{n-k}}{\binom{2n}{n}} \tag{2}$$

We use proof certificate

$$R(n,k) = \frac{2k-1}{2n+1}, \tag{3}$$

which is the same as using

$$G(n,k) = \frac{2k-1}{2n+1} \frac{(-1)^{k-1} \cdot \binom{n}{k-1}\binom{2(k-1)}{k-1} 4^{n-(k-1)}}{\binom{2n}{n}}, \tag{4}$$

the automatic solver has verified that

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k). \tag{5}$$

Thereafter user now has to verify that

$$\lim_{k\to\pm\infty} G(n,k) = 0 \ \ \forall\, n. \tag{6}$$

Then we get

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0 \tag{7}$$

Lastly equation 1 needs to be verified for some n, for instance $n = 0$. Thereafter the identity is shown.

For $n = 0$ we get

$$\sum_k (-1)^k \binom{n}{k}\binom{2k}{k} 4^{n-k} = (-1)^0 \binom{0}{0}\binom{0}{0} 4^0 = 1$$

Also $\binom{0}{0} = 1$.

Results as an example

$k < 1$ and
$k > n+1$
gives that

$$\binom{n}{k-1} = 0,$$

$$\implies$$

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

Proof

Automatic WZ-method prover

2019-11-25

We want to prove that

$$\sum (-1)^k \cdot \binom{n}{k}\binom{2k}{k}4^{n-k} = \binom{2n}{n} \tag{1}$$

holds. By dividing equation 1 by the right hand side we get

$$F(n,k) = \frac{(-1)^k \cdot \binom{n}{k}\binom{2k}{k}4^{n-k}}{\binom{2n}{n}} \tag{2}$$

We use proof certificate

$$R(n,k) = \frac{2k-1}{2n+1}, \tag{3}$$

which is the same as using

$$G(n,k) = \frac{2k-1}{2n+1}\frac{(-1)^{k-1} \cdot \binom{n}{k-1}\binom{2(k-1)}{k-1}4^{n-(k-1)}}{\binom{2n}{n}}, \tag{4}$$

the automatic solver has verified that

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k). \tag{5}$$

Thereafter user now has to verify that

$$\lim_{k\to\pm\infty} G(n,k) = 0 \ \ \forall\, n. \tag{6}$$

Then we get

$$\sum_k F(n+1,k) - F(n,k) = \sum_k G(n,k+1) - G(n,k) = 0 \tag{7}$$

Lastly equation 1 needs to be verified for some n, for instance $n = 0$. Thereafter the identity is shown.

For $n = 0$ we get

$$\sum_k (-1)^k \binom{n}{k}\binom{2k}{k}4^{n-k} =$$

$$(-1)^0\binom{0}{0}\binom{0}{0}4^0 = 1$$

Also $\binom{0}{0} = 1$.

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

DOES NOT WORK

- Cannot come up with solution, only prove
- Some parts are left for the user
- Similar examples with different results

WORKS WELL

- Solves most examples
- Gives a solution quickly

Future work

- Can Wilf-Zeilberger's method be used on other types of problems? (not binomial coefficients)
- Combine the program with guessing solution to identity
- Computer algebra in general

Future work

- Can Wilf-Zeilberger's method be used on other types of problems? (not binomial coefficients)
- Combine the program with guessing solution to identity
- Computer algebra in general

Future work

- Can Wilf-Zeilberger's method be used on other types of problems? (not binomial coefficients)
- Combine the program with guessing solution to identity
- Computer algebra in general

Future work

- Can Wilf-Zeilberger's method be used on other types of problems? (not binomial coefficients)
- Combine the program with guessing solution to identity
- Computer algebra in general

Thank you for listening!

Polynomials – Representation

Polynomial

$$p(k) = a_0 + a_1 k + \ldots + a_m k^m$$

is stored as

$$[a_0, a_1, \ldots, a_m].$$

Polynomials 1 – Example

The polynomial

$$p(k, m) = 1 + k^2 + km - m^2 + km^2 + k^2m^2$$

is stored as

$$\Big[[1, 0, 1], [0, 1, 0], [-1, 1, 1]\Big].$$

Polynomials – Addition

Assume we want to add $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$. Then we get $h = [h_0, \ldots, h_m]$ where $m = max(m_f, m_g)$. Then we have that

$$h_i = f_i + g_i,$$

if f_i and g_i are integers. Otherwise we get

$$h_i = ADD(f_i, g_i).$$

Polynomials – Addition

Assume we want to add $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = max(m_f, m_g)$.Then we have that

$$h_i = f_i + g_i,$$

if f_i and g_i are integers.Otherwise we get

$$h_i = ADD(f_i, g_i).$$

Polynomials – Addition

Assume we want to add $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = max(m_f, m_g)$.Then we have that

$$h_i = f_i + g_i,$$

if f_i and g_i are integers.Otherwise we get

$$h_i = ADD(f_i, g_i).$$

Polynomials – Addition

Assume we want to add $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = max(m_f, m_g)$.Then we have that

$$h_i = f_i + g_i,$$

if f_i and g_i are integers.Otherwise we get

$$h_i = ADD(f_i, g_i).$$

Polynomials – Multiplication

Assume we want to multiply $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$. Then we get $h = [h_0, \ldots, h_m]$ where $m = m_f + m_g$. Then we have that

$$h_i = \sum_{k=0}^{i} f_k \cdot g_{i-k},$$

if f_i and g_i are of one and the same variable. Otherwise we get

$$h_i = \sum_{k=0}^{i} MULTIPLY(f_k, g_{i-k}).$$

Polynomials – Multiplication

Assume we want to multiply $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = m_f + m_g$.Then we have that

$$h_i = \sum_{k=0}^{i} f_k \cdot g_{i-k},$$

if f_i and g_i are of one and the same variable.Otherwise we get

$$h_i = \sum_{k=0}^{i} MULTIPLY(f_k, g_{i-k}).$$

Polynomials – Multiplication

Assume we want to multiply $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = m_f + m_g$.Then we have that

$$h_i = \sum_{k=0}^{i} f_k \cdot g_{i-k},$$

if f_i and g_i are of one and the same variable.Otherwise we get

$$h_i = \sum_{k=0}^{i} MULTIPLY(f_k, g_{i-k}).$$

Polynomials – Multiplication

Assume we want to multiply $f = [f_0, \ldots, f_{m_f}]$ and $g = [g_0, \ldots, g_{m_g}]$.Then we get $h = [h_0, \ldots, h_m]$ where $m = m_f + m_g$.Then we have that

$$h_i = \sum_{k=0}^{i} f_k \cdot g_{i-k},$$

if f_i and g_i are of one and the same variable.Otherwise we get

$$h_i = \sum_{k=0}^{i} MULTIPLY(f_k, g_{i-k}).$$

Polynomials – Division

Usually division (a divided by b) is done by finding polynomials q, r such that

$$a = q \cdot b + r,$$

and $deg(r) < deg(b)$. This is not possible in integer coefficients. Therefore we use q, r, f such that

$$f \cdot a = q \cdot b + r,$$

$deg(r) < deg(b)$ and f has the same variable setup as the coefficients of a and b.

Polynomials – Division

Usually division (a divided by b) is done by finding polynomials q, r such that

$$a = q \cdot b + r,$$

and $deg(r) < deg(b)$. This is not possible in integer coefficients. Therefore we use q, r, f such that

$$f \cdot a = q \cdot b + r,$$

$deg(r) < deg(b)$ and f has the same variable setup as the coefficients of a and b.

Polynomials – Division

Usually division (a divided by b) is done by finding polynomials q, r such that

$$a = q \cdot b + r,$$

and $deg(r) < deg(b)$.This is not possible in integer coefficients.Therefore we use q, r, f such that

$$f \cdot a = q \cdot b + r,$$

$deg(r) < deg(b)$ and f has the same variable setup as the coefficients of a and b.

Polynomials – GCD

We get *gcd* by Euclid's algorithm. With division as

$$a = q \cdot b + r$$

gcd is usually done by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } gcd(b, r).$$

With division as

$$f \cdot a = q \cdot b + r$$

we get *gcd* by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } \frac{g}{\bar{g}} gcd(\bar{a}, \bar{b}),$$

where $\bar{x}$ denotes *gcd* of the coefficients in x and $g = gcd(b, r)$.

Polynomials – GCD

We get *gcd* by Euclid's algorithm.With division as

$$a = q \cdot b + r$$

gcd is usually done by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } gcd(b, r).$$

With division as

$$f \cdot a = q \cdot b + r$$

we get *gcd* by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } \frac{g}{\bar{g}} gcd(\bar{a}, \bar{b}),$$

where $\bar{x}$ denotes *gcd* of the coefficients in x and $g = gcd(b, r)$.

Polynomials – GCD

We get *gcd* by Euclid's algorithm.With division as

$$a = q \cdot b + r$$

gcd is usually done by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } gcd(b, r).$$

With division as

$$f \cdot a = q \cdot b + r$$

we get *gcd* by

$$gcd(a, b) = a \text{ if } b = 0 \text{ else } \frac{g}{\bar{g}} gcd(\bar{a}, \bar{b}),$$

where $\bar{x}$ denotes *gcd* of the coefficients in x and $g = gcd(b, r)$.

Proof generation

Steps of proof generation

- Get input and parser
- Parse input → get $F(n, k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n, k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Proof generation

Steps of proof generation

- Get input and parser
- Parse input $\rightarrow$ get $F(n, k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n, k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Proof generation

Steps of proof generation

- Get input and parser
- Parse input $\rightarrow$ get $F(n,k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n,k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Proof generation

Steps of proof generation

- Get input and parser
- Parse input $\rightarrow$ get $F(n,k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n,k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Proof generation

Steps of proof generation

- Get input and parser
- Parse input $\rightarrow$ get $F(n, k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n, k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Proof generation

Steps of proof generation

- Get input and parser
- Parse input $\rightarrow$ get $F(n,k)$ and $\frac{a_k}{a_{k-1}}$
- Get $G(n,k)$ from Gosper's algorithm
- Write proof in LaTeXformat
- Highlight parts that the user need to complete

Dependencies of the code