Wilf Zeilberger method

Lars Åström

LUND
UNIVERSITY
Department of Mathematics

MSc Thesis TFRT-TODO
ISSN 0280–5316

Department of Mathematics
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

Preface

Contents

1

Introduction

2

Background

3

Theory

3.1 Polynomials

In the thesis we are continuously discussing polynomials. We are always using polynomials with integer coefficients – which we also generalize to include rational coefficients by looking at fractions of polynomials with integer coefficients. Therefore, from now on whenever we talk about polynomials they are assumed to have integer coefficients as long as otherwise is not stated.

We will now have a discussion on what operations we want to be able to perform on the polynomials both theoreticly and partly implementation wise. Therefore there will not be a specific subchapter regarding polynomials later on in chapter 4. But first we need to describe how a polynomial can be implemented and stored in the code – and this is how it was implemented in this thesis.

A polynomial, say $p(n)$, can be seen as a list of coefficients $[a_0, a_1, \dots, a_d]$. This represents the polynomial

$$p(n) = a_0 + a_1 n + \ldots + a_d n^d. \tag{3.1}$$

This is fairly simple if the polynomial only has one variable. As soon as we introduce polynomials with more than one variable – which is the case in all examples where Wilf-Zeilbergers method is used – it gets a bit more complicated. The way we will visualize polynomials of more than one variable is to try to keep the very same structure as in 3.1, namely we store a polynomial $p(n)$ as a list of coefficients $[a_0, a_1, \dots, a_d]$, but instead of having a_i just representing integers they are instead themselves polynomials. In order to show how this can be done, lets show an example.

EXAMPLE 3.1 Lets look at the polynomial

$$\begin{aligned} p(k,m,n) =& 1+3n+2n^2-m-2mn-2mn^2+3m^2+3m^2n+3m^2n^2+ \\ & 3k-3kn+2kn^2+km-2kmn-km^2+2k^2-k^2n-k^2n^2+ \\ & 3k^2m-3k^2mn-2k^2mn^2-3k^2m^2-k^2m^2n-3k^2m^2n^2. \end{aligned}$$

This will be stored as

$$\begin{aligned} p(k,m,n) =& -((3k^2+3k-3)m^2-(2k+1)m-(3k^2-k+3))n^2+ \\ & ((k^2-1)m^2-(k^2-2k+2)m-(2k+2))n+ \\ & ((2k^2+2)m^2+(2k-1)m+(k^2-k-3)) \end{aligned}$$

Note that this can be written as

$$p(k,m,n) = p_2(k,m)n^2 + p_1(k,m)n + p_0,$$

where

$$\begin{aligned} p_0(k,m) &= ((2k^2+2)m^2+(2k-1)m+(k^2-k-3)) \\ p_1(k,m) &= ((k^2-1)m^2-(k^2-2k+2)m-(2k+2)) \\ p_2(k,m) &= -((3k^2+3k-3)m^2-(2k+1)m-(3k^2-k+3)). \end{aligned}$$

Now since our definition of polynomials is recursive, each of $p_0(k,m), p_1(k,m), p_2(k,m)$ is written as $q_2(k)m^2 + q_1(k)m + q_0(k)$ for polynomials q_0, q_1, q_2. For instance for p_1 we have

$$p_1(k,m) = q_2(k)m^2 + q_1(k)m + q_0(k),$$

where

$$\begin{aligned} q_0(k) &= -(2k+2) \\ q_1(k) &= -(k^2-2k+2) \\ q_2(k) &= (k^2-1) \end{aligned}$$

Lastly we have the polynomials q_0, q_1, q_2 which have integers as coefficients. □

As we saw in example 3.1 we define all polynomials as in equation 3.1 where the coefficients a_i are polynomials or integers, where it is only integers in the base case.

Note that in this case we wrote the polynomial with n ”outmost” in the representation of p. There was no specific reason for this, and the choice is not very important when only looking at one polynomial at a time. When we are using several polynomials at the same time, for instance if we want to add two polynomials, the structure is crucial. Therefore we will define some concepts regarding the representation of a polynomial.

DEFINITION 3.1 A polynomial p is said to have the variable order $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ if it is stored as

$$p(\mathbf{x}) = \sum_{i_1=0}^{d_1} \left(\sum_{i_2=0}^{d_2} \left(\sum_{i_3=0}^{d_3} \ldots \right) x_2^{i_2} \right) x_1^{i_1}. \tag{3.2}$$

Note that the same polynomial can be expressed in various different ways, and we say that two variable orders, $\mathbf{x}$ and $\mathbf{y}$, are different if $|\mathbf{x}| \neq |\mathbf{y}|$ (where $|\mathbf{z}|$ denotes the length of the vector $\mathbf{z}$) or there exist an integer i such that $x_i \neq y_i$. □

REMARK If the polynomial p has variable order $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, then the coefficients of p have variable order $\mathbf{x}' = (x_2, \ldots, x_m)$. □

REMARK Note that if the polynomial p has variable order $\mathbf{x}$ such that $|\mathbf{x}| = 1$, then the coefficients of p are integers. □

Here we can note that d_i is the degree of p with respect to x_i. Furthermore we note that we can express a polynomial p with the variable order $\mathbf{x} = (x_1, \ldots, x_m)$ even though p does not have all the variables x_j in its expression. If p does not depend on x_j that will be seen in that $d_j = 0$. A polynomial p cannot, however, be expressed with the variable order $\mathbf{x}$ if not all its variables are in $\mathbf{x}$. A last note is that we can convert a polynomial p with variable order $\mathbf{x}$ to any variable order $\mathbf{y}$ as long as $x_i \in \mathbf{y} \;\; \forall\, x_i \in \mathbf{x}$.

There are of course many operations that are needed when operating with polynomials; addition, negate, subtraction, multiplication, division, modulo, gcd (greatest common divisor), checking for equality, finding roots and evaluating at a point. In the implementation all of them are defined recursively. We will show how this is done for all the most straight forward operations here, while the more complicated operations – division, modulo and gcd – are described later in section 3.1.1.

First we define how we write that a polynomial is constructed:

DEFINITION 3.2 When we construct a polynomial p of a variable x then we write this as

$$p = polynomial(\mathbf{a}, "x"), \tag{3.3}$$

where $\mathbf{a} = (a_0, a_1, \ldots, a_d)$. This means that

$$p(x) = a_0 + a_1 x + \ldots + a_d x^d. \tag{3.4}$$

Note that we have not specified what a_i are; they can be either integers or polynomials – but if they are polynomials they have to have the same variable order. Also note that once we have the polynomial p we will use the notation $p[i]$ to reference the i-th variable, namely a_i. If we reference $p[j]$ where $j \geq d$, then this is defined as $p[j] = 0 \;\; \forall\, j \geq d$. □

We also define the degree of a polynomial:

DEFINITION 3.3 Let the polynomial p have variable order $\mathbf{x} = (x_1, \ldots, x_m)$. Then we define the degree of p, denoted $deg(p)$, as the largest exponent of x_1 in the expression for p. □

For all the operations we will use two polynomials f and g. These are polynomials with the variable order $\mathbf{x}$. We will now provide procedures that show how all the previously mentioned operations should work. Note that the procedures are assuming that the polynomials have the same variable order, if this is not the case then the polynomials first are converted into a common variable order.

All the procedures try to follow the same structure, which is quite recursive. All procedures have a base case, usually when $|\mathbf{x}| = 1$, and in all other cases recursive calls are made. A note to be made here is that recursive calls are not computationally heavy in theory but might take time if many are made, especially if the depth is large, in practice. In this thesis however, the depth is very small – usually less than 5 – since the depth comes from the number of variables in the polynomials. Therefore the recursion is not expected to cause any problems computationally.

Algorithm 1 Addition

```
    Input: Polynomials $f$ and $g$, with variable order $\mathbf{x}$
    Output: Polynomial $h$ such that $h = f + g$

procedure ADD($f, g$)
    $\mathbf{a} \leftarrow [\,]$
    for $i \leftarrow 0, \ldots, max(deg(f), deg(g))$ do
        if $|\mathbf{x}| = 1$ then
            $\mathbf{a}[i] \leftarrow f[i] + g[i]$
        else
            $\mathbf{a}[i] \leftarrow$ ADD($f[i], g[i]$)
        end if
    end for
    return $polynomial(\mathbf{a}, \mathbf{x}[0])$
end procedure
```

Algorithm 2 Negatation

```
Input: Polynomial $f$ with variable order $\mathbf{x}$
Output: Polynomial $h$ such that $h = -f$

procedure Negate($f$)
    $\mathbf{a} \leftarrow [\ ]$
    for $i \leftarrow 0, \ldots, deg(f)$ do
        if $|\mathbf{x}| = 1$ then
            $\mathbf{a}[i] \leftarrow -f[i]$
        else
            $\mathbf{a}[i] \leftarrow$ Negate($f[i]$)
        end if
    end for
    return $polynomial(\mathbf{a}, \mathbf{x}[0])$
end procedure
```

Algorithm 3 Subtraction

```
Input: Polynomials $f$ and $g$, with variable order $\mathbf{x}$
Output: Polynomial $h$ such that $h = f - g$

procedure Subtract($f, g$)
    $g' \leftarrow$ Negate($g$)
    return Add($f, g'$)
end procedure
```

Algorithm 4 Multiplication

```
Input: Polynomials $f$ and $g$, with variable order $\mathbf{x}$
Output: Polynomial $h$ such that $h = f \cdot g$

procedure MULTIPLY($f, g$)
    $\mathbf{a} \leftarrow [\,]$
    for $i \leftarrow 0, \ldots, deg(f) + deg(g)$ do
        $\mathbf{a}[i] \leftarrow 0$
    end for
    for $i \leftarrow 0, \ldots, deg(f)$ do
        for $j \leftarrow 0, \ldots, deg(g)$ do
            if $|\mathbf{x}| = 1$ then
                $\mathbf{a}[i+j] \leftarrow \mathbf{a}[i+j] + f[i] \cdot g[j]$
            else
                $y_{i,j} \leftarrow$ MULTIPLY($f[i], g[j]$)
                $\mathbf{a}[i+j] \leftarrow$ ADD($\mathbf{a}[i+j], y_{i,j}$)
            end if
        end for
    end for
    return $polynomial(\mathbf{a}, \mathbf{x}[0])$
end procedure
```

Algorithm 5 Equality

```
Input: Polynomials $f$ and $g$, with variable order $\mathbf{x}$
Output: $True$ if $f$ and $g$ are equal, otherwise $False$

procedure EQUALS($f, g$)
    if $deg(f) \neq deg(g)$ then
        return $False$
    end if
    for $i \leftarrow 0, \ldots, deg(f)$ do
        if $|\mathbf{x}| = 1$ AND $f[i] \neq g[i]$ then
            return $False$
        else if $|\mathbf{x}| > 1$ AND $not$ EQUALS($f[i], g[i]$) then
            return $False$
        end if
    end for
    return $True$
end procedure
```

Algorithm 6 Evaluating at a point

```
Input: Polynomials $f$, with variable order $\mathbf{x}$, variable $v \in \mathbf{x}$ and value $y$
Output: Polynomial $h$ that is $f$ evaluated at $v = y$

1: procedure EVALUATE($f, v, y$)
2:     if $\mathbf{x}[0] = v$ then
3:         $g \leftarrow f[0]$
4:         for $i \leftarrow 1, \dots, deg(f)$ do
5:             $h \leftarrow$ MULTIPLY($f[i], y^i$)
6:             $g \leftarrow$ ADD($g, h$)
7:         end for
8:         return $g$
9:     else
10:        $\mathbf{a} \leftarrow [\ ]$
11:        for $i \leftarrow 0, \dots, deg(f)$ do
12:            $\mathbf{a}[i] \leftarrow$ EVALUATE($f[i], v, y$)
13:        end for
14:        return $polynomial(\mathbf{a}, \mathbf{x}[0])$
15:    end if
16: end procedure
```

The next procedure is used to find roots of a polynomial. Here if the polynomial is only dependent on one variable, meaning $|\mathbf{x}| = 1$, then we solve this equation (we call this function *SolveEquation*). In the implementation of the program only solving equations up to degree 2 is implemented, but the program also tries to insert values in the range $(-100, 100)$ and sees if any of these are roots. We do not include the procedure for *SolveEquation* in the report, since it is just solving a second degree equation and additionally trying a bunch of different small values.

Algorithm 7 Finding roots

Input: Polynomial f, with variable order $\mathbf{x} = (x_1, \dots, x_m)$
Output: Integers x' such that $x_m = x' \implies f = 0$

```
procedure ROOTS($f$)
    if $|\mathbf{x}| = 1$ then
        return SOLVEEQUATION($f$)
    else
        $\mathbf{a} \leftarrow$ ROOTS($f[0]$)
        for $i \leftarrow 1, \dots, deg(f)$ do
            $\mathbf{b} \leftarrow [\,]$
            for all $a' \in \mathbf{a}$ do
                if EVALUATE($f[i], x_m, a'$)=0 then
                    $\mathbf{b}[size(\mathbf{b})] \leftarrow a'$
                end if
            end for
            $\mathbf{a} \leftarrow \mathbf{b}$
        end for
        return $\mathbf{a}$
    end if
end procedure
```

3.1.1 Division, modulo and GCD

Now we come to the slightly more complicated operations that we need to perform on polynomials. First we define GCD for polynomials a bit more carefully, which we do in a few steps.

DEFINITION 3.4 A polynomial g is said to divide another polynomial a if there exists a polynomial q such that $a = g \cdot q$. We will use the notation $g|a$ to denote that g divides a. □

DEFINITION 3.5 A polynomial g is called a *common divisor* to two polynomials a and b if $g|a$ and $g|b$. □

DEFINITION 3.6 We say that a polynomial g is the *greatest common divisor* (GCD) to two polynomials if

1. g is a common divisor to a and b
2. For all common divisors g' to a and b we have that $g'|g$. □

In general performing division, modulo and gcd is fairly straight forward even on polynomials. Usually one might have the following structure on the implementation:

$$DIVIDE(a,b) \to (q,r) \text{ such that } a = b\cdot q + r, deg(r) < deg(b), \tag{3.5}$$

$$MODULO(a,b) \to r \text{ where } DIVIDE(a,b) = (q,r), \tag{3.6}$$

$$GCD(a,b) \to a \text{ if } b = 0 \text{ otherwise } GCD(b, MODULO(a,b)), \tag{3.7}$$

where $\to$ indicates what the function returns.

This implementation scheme is fairly straight forward, even though the divide function might be a bit messy. The problem we encounter in this thesis though, is that we cannot necessarily find polynomials q,r such that 3.5 is satisfied. The reason for this is that we are working with integer coefficients and not in a ring. In order to show the problems we encounter we will use a common example throughout this section to show what the problem is and how we solve it.

EXAMPLE 3.2 Consider dividing $a(x) = x^2+2x+1$ by $b(x) = 2x$. Assume we have q,r that fulfill 3.5. Then we need to have $deg(r) < deg(b) = deg(2x) = 1$. This means that $r(x) = c$, where c is an integer constant. This gives us

$$x^2+2x+1 = (2x)q(x)+c. \tag{3.8}$$

Furthermore we have that $deg(\text{Left hand side}) = 2$, hence $deg(q) = 1$. Let $q(x) = c_0 + c_1x$, then 3.8 becomes

$$x^2+2x+1 = (2x)(c_1x+c_0)+c = 2c_1x^2+2c_0x+c. \tag{3.9}$$

By looking at degree 2 we get $2c_1 = 1$, which of course does not have any solutions for integers c_1. □

The way we solve this is to instead of giving two polynomials q,r when we divide a by b we also give a third parameter, f, which is a factor of the same type as the coefficients in a and b. The condition that needs to be fulfilled is

$$DIVIDE(a,b) \to (q,r,f) \text{ such that } f\cdot a = b\cdot q + r, deg(r) < deg(b). \tag{3.10}$$

Now we will show that it actually is possible to find q,r,f such that 3.10 holds.

THEOREM 3.1 Let $a(x_1), b(x_1)$ be polynomials of the same variable order $\mathbf{x} = (x_1,\dots,x_m)$. Then it is possible to find polynomials q,r,f, where q,r have variable order $\mathbf{x}$ and f has variable order $\mathbf{x}' = (x_2,\dots,x_m)$, such that

$$f\cdot a(x_1) = q(x_1)b(x_1)+r(x_1), \tag{3.11}$$

and $deg(r) < deg(b)$. □

Proof Let $a = a_0 + a_1x_1 + \ldots + a_cx_1^c$ and $b = b_0 + b_1x_1 + \ldots + b_dx_1^d$, where c,d are the degrees of a and b, respectively. If $c < d$ then we can use $q = 0, r = a, f = 1$ and we have that 3.11 is fulfilled.

Now assume that $c \geq d$. We will now show that we can find a', q_0, f_0 such that

$$f_0a(x_1) = q_0(x_1)b(x_1) + a'(x_1). \tag{3.12}$$

This is obtained by choosing $f_0 = b_d, q_0 = a_cx^{c-d}$ and $a'(x_1) = b_da(x_1) - a_cx^{c-d}b(x_1)$. This gives us

$$\begin{aligned} a'(x_1) = b_da(x_1) - a_cx^{c-d}b(x_1) = b_d(a_0 + \ldots + a_cx^c) - a_cx^{c-d}(b_0 + \ldots + b_dx^d) \\ = b_d(a_0 + \ldots + a_{c-1}x^{c-1}) - a_cx^{c-d}(b_0 + \ldots + b_{d-1}x^{d-1}) \end{aligned} \tag{3.13}$$

Now we can clearly see that $deg(a') < c$, which means we have reduced the problem by (at least) 1 in the degree of a. We continue to do this until the degree of a is less than the degree of b, and then we collect the results. If we have that

$$DIVIDE(a', b) \to (q', r', f'), \tag{3.14}$$

then since a' is given by 3.13 we get

$$\begin{aligned} f' \cdot a' = q' \cdot b + r' & \qquad \Longrightarrow \\ f'(b_da - a_cx^{c-d}b) = q' \cdot b + r' & \qquad \Longrightarrow \\ (fb_d)a = (q' + a_cx^{c-d})b + r', & \end{aligned} \tag{3.15}$$

and thus we get

$$DIVIDE(a, b) \to (q, r, f) = (q' + a_cx^{c-d}, r', fb_d). \tag{3.16}$$

Now we have shown the theorem, and provided the method for dividing polynomials at once. □

DEFINITION 3.7 When we divide polynomial a by polynomial b (both having variable order $\mathbf{x} = (x_1, \ldots, x_m)$) we are given three things:

- q, a polynomial with variable order $\mathbf{x}$, which is called the ”quotient”,
- r, a polynomial with variable order $\mathbf{x}$, which is called the ”remainder”,
- f, a polynomial with variable order $(x_2, \ldots, x_m)$, which is called the ”factor”.

These polynomials fulfill equation 3.10. □

REMARK One important thing to note is that the definition of how we divide is not unique, if q, r, f are all multiplied by a polynomial s with variable order $(x_2, \dots, x_m)$ then equation 3.10 will still be fulfilled. $\square$

Now we have defined division and will soon move over to calculating the greatest common divisor, but first we have a look at what happens to our example.

EXAMPLE 3.3 Consider dividing $a(x) = x^2 + 2x + 1$ by $b(x) = 2x$. Now we see that $q = x + 2, r = 2, f = 2$ gives us a solution where $0 = deg(r) < deg(b) = 1$. $\square$

We will show a theorem which states how the greatest common divisor is computed, but first we need a little bit of notation.

DEFINITION 3.8 Let p be a polynomial with variable order $\mathbf{x} = (x_1, \dots, x_m)$. If we write p as $p(x_1) = p_0 + p_1 x_1 + \dots p_d x_1^d$, then we denote the greatest common divisor of all the coefficients $p_i, 0 \le i \le d$ by $gcd_c(p)$. $\square$

THEOREM 3.2 Let g denote the greatest common divisor of polynomials a and b. Then we can compute this by the following:

1. If $b = 0$, then $g = a$,
2. Otherwise g is given by

$$g = \frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b)), \tag{3.17}$$

   where g' is the greatest common divisor of b and r, where r is the remainder when a is divided by b, which means

$$DIVIDE(a, b) \to (q, r, f). \tag{3.18}$$

$\square$

Before we prove theorem 3.2 we show the reasoning behind why we get this more complicated version of gcd in 3.17 instead of the much easier version in 3.7. We start by showing the example and see what happens if we naïvely using the new division combined with 3.7.

EXAMPLE 3.4 Consider computing the greatest common divisor of $a(x) = x^2 + 2x + 1$ and $b(x) = 2x$ using 3.7.

$$gcd(x^2 + 2x + 1, 2x) = gcd(2x, 2) = gcd(2, 0) = 2, \tag{3.19}$$

since

$$\begin{aligned} DIVIDE(x^2+2x+1,2x) &\to (q,r,f) = (x+2,2,2) \\ DIVIDE(2x,2) &\to (q,r,f) = (x,0,1). \end{aligned} \tag{3.20}$$

But now we see that we have the greatest common divisor being 2, but 2 does not divide $a(x)$. This highlights a problem with using 3.7 for gcd.

Now let us consider using 3.17 instead. This gives us (working backwards in steps)

$$\begin{aligned} gcd(2,0) &= 2 \\ gcd(2x,2) &= \frac{gcd(2,0)}{gcd_c(gcd(2,0))} \cdot gcd(gcd_c(2x), gcd_c(2)) = \frac{2}{2} \cdot gcd(2,2) = 2 \\ gcd(x^2+2x+1,2x) &= \frac{gcd(2x,2)}{gcd_c(gcd(2x,2))} \cdot gcd(gcd_c(x^2+2x+1), gcd_c(2x)) = \\ &= \frac{2}{2} \cdot gcd(1,2) = 1. \end{aligned} \tag{3.21}$$

Using 3.17 seems to work on this example. We need, however to prove it in general and to prove it regardless of which q,r,f are used for $DIVIDE(a,b)$. □

As we saw in example 3.4 if we try to naïvely use 3.7 to compute the gcd, but with the new definition of how we divide, then that results in a too large divisor since the factor f allows the divisor to be larger. This can be seen by looking at the formula for dividing a by b. We have that

$$f \cdot a(x_1) = b(x_1) \cdot q(x_1) + r(x_1). \tag{3.22}$$

Now the greatest common divisor of b and r will divide the right hand side $f \cdot a(x_1)$ but not necessarily $a(x_1)$. With this insight in mind, we will now prove theorem 3.2, but first we need a few lemmas.

LEMMA 3.1 Let a,b,d be polynomials such that $d|(a \cdot b)$. Then we can factor d into two polynomials d_a, d_b such that

1. $d = d_a \cdot d_b$
2. $d_a|a, d_b|b$ □

Proof Let $d_a = gcd(a,d)$. Then we get that $d_a|a, d_a|d \iff \exists\, a', d' : d = d_a d', a = d_a a'$ and $gcd(a',d') = 1$. We know that $d|(a \cdot b) \iff \exists\, x : a \cdot b = d \cdot x$. Now we replace a and d by $d_a a', d = d_a d'$ which gives us $a' \cdot b = d' \cdot x$, where $gcd(a',d') = 1$. Therefore $d'|b$, and thus we can choose $d_b = d'$. □

LEMMA 3.2 Let p be a polynomial $p(x_1) = p_0 + \ldots + p_d x_1^d$ with variable order $\mathbf{x} = (x_1, \ldots, x_m)$ and f be a polynomial with variable order $(x_2, \ldots, x_m)$ such that $f|p$. Then we have that $f|p_i \;\; \forall\, 0 \leq i \leq d$. □

Proof We know that $f|p$, which is equivalent to $p = fp'$ for some polynomial p'. Let $p' = p'_0 + \ldots + p'_c x_1^c$. Now first of all, since f does not depend on x_1 we have that $deg(p) = deg(p') = d$. Furthermore we have that

$$p_0 + p_1 x_1 + \ldots + p_d x_1^d = p = fp = f(p'_0 + p'_1 x_1 + \ldots + p'_d x_1^d) = \\ = (fp'_0) + (fp'_1)x_1 + \ldots + (fp'_d)x_1^d \qquad (3.23)$$

Now by looking at the terms for each degree of x_1 in 3.23 we see that

$$p_i = fp'_i \iff f|p_i, \;\; \forall\, 0 \leq i \leq d. \qquad (3.24)$$

□

LEMMA 3.3 Let p and d be polynomials with variable order $\mathbf{x} = (x_1, \ldots, x_m)$, $gcd_c(d) = 1$ and f be a polynomial with variable order $(x_2, \ldots, x_m)$ such that $d|p$ and $f|p$. Then we have that $(d \cdot f)|p$. □

Proof We know that $f|p$, hence we can write p as $p = fp'$. We know that $d|p = fp'$ and thus we can due to lemma 3.1 factor g into $d_f, d_{p'}$ such that $d = d_f \cdot d_{p'}$ and $d_f|f, d_{p'}|p$. But since f has variable order $(x_2, \ldots, x_m)$ then d_f cannot depend on x_1 and thus can be expressed with variable order $(x_2, \ldots, x_m)$. Furthermore $d_f|d$, which together with lemma 3.2 gives us that $d_f|d_i$ for all coefficients d_i of d. Therefore $d_f|gcd_c(d) = 1 \implies d_f = 1$. Thus $d = d_{p'}|p'$ and therefore $(d \cdot f)|p$ □

Now we are ready to finish this section with the proof of theorem 3.2.

Proof Theorem 3.2 Recall that $gcd(b, r)$ is denoted by g'. We will show this theorem in two steps. First of all we want to prove that

$$g = \frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b)), \qquad (3.25)$$

in fact is a divisor of a and b. After that we will show that

$$d|a, d|b \implies d \left| \frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b)). \right. \qquad (3.26)$$

We start with the first part. Since $g'|b$ and $g'|r$, g' divides the right hand side of 3.22. But the left hand side is equal to the left hand side, thus we know that $g'|fa$. This in turns implies that we (due to lemma 3.1) can factor g into two parts, g_f, g_a, such that $g' = g_a g_f$, $g_a|a$ and $g_f|f$. But now since f has variable order $(x_2,\dots,x_m)$ then g_f cannot depend on x_1.

We will now show that $\frac{g'}{gcd_c(g')}|a$. We know that since g_f is of variable order $(x_2,\dots,x_m)$ and $g_f|g'$, then by lemma 3.2 we get that $g_f|g'_i$ for all coefficients g'_i in g'. This means that $g_f|gcd_c(g')$. We also know that

$$\frac{g'}{gcd_c(g')} \cdot gcd_c(g') = g' = g_f \cdot g_a, \tag{3.27}$$

therefore we get that $\frac{g'}{gcd_c(g')}|g_a$, and thus $\frac{g'}{gcd_c(g')}|a$. Furthermore

$$gcd_c\Big(\frac{g'}{gcd_c(g')}\Big) = 1. \tag{3.28}$$

We also clearly see that $gcd(gcd_c(a), gcd_c(b))|gcd_c(a)$ and $gcd_c(a)|a$. Therefore $gcd(gcd_c(a), gcd_c(b))$ divides a. Now we use lemma 3.3 and conclude that with

$$\begin{aligned} &p = a, \\ &d = \frac{g'}{gcd_c(g')}, \\ &f = gcd(gcd_c(a), gcd_c(b)) \end{aligned} \tag{3.29}$$

plugged into lemma 3.3 we get that

$$\frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b))|a \tag{3.30}$$

Furthermore we see that

$$\frac{g'}{gcd_c(g')}\bigg|g', g'|b \implies \frac{g'}{gcd_c(g')}\bigg|b. \tag{3.31}$$

Therefore, again according to lemma 3.3 we get that

$$\frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b))|b. \tag{3.32}$$

This finished the first part of the proof. Now we need to show that

$$d|a, d|b \implies d\bigg|\Big(\frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b))\Big), \tag{3.33}$$

where d is a polynomial with variable order $\mathbf{x} = (x_1,\dots,x_m)$.

First we note that any polynomial that divides a and b also divide r, which is clear by looking at equation 3.22. We now factor d into $d_1 \cdot d_2$, where $d_1 = \frac{d}{gcd_c(d)}, d_2 = gcd_c(d)$. We will now show that $d_1 | \frac{g'}{gcd_c(g')}$ and $d_2 | gcd(gcd_c(a), gcd_c(b))$.

We have that $d_1 | d, d | g' \implies d_1 \Big| \frac{g'}{gcd_c(g')} \cdot gcd_c(g')$. Now lemma 3.1 gives us that d_1 can be factored into d_{11}, d_{12} such that $d_{11} \Big| \frac{g'}{gcd_c(g')}$ and $d_{12} | gcd_c(g')$. But $d_{12} | gcd_c(g')$ means that d_{12} has variable order $(x_2, \ldots, x_m)$ But this gives us that $d_{12} = 1$ since $d_1 = \frac{d}{gcd_c(d)}$ (otherwise we would have a larger common divisor for the coefficients of d than $gcd_c(d)$). Hence $d_1 = d_{11} \Big| \frac{g'}{gcd_c(g')}$.

Next we show that $d_2 | gcd(gcd_c(a), gcd_c(b))$. We know that $d_2 | d, d | a, d | b \implies d_2 | a, d_2 | b$. Furthermore d_2 has variable order $(x_2, \ldots, x_m)$, therefore lemma 3.2 gives us that $d_2 | a_i$ for all coefficients a_i in a, and $d_2 | b_i$ for all coefficients b_i in b. This gives us that $d_2 | gcd_c(a), d_2 | gcd_c(b)$, therefore d_2 is a common divisor of $gcd_c(a), gcd_c(b)$, hence $d_2 | gcd(gcd_c(a), gcd_c(b))$.

This finished the proof for that the greatest common divisor of polynomials a, b is

$$g = \frac{g'}{gcd_c(g')} \cdot gcd(gcd_c(a), gcd_c(b)), \tag{3.34}$$

where $g' = gcd(b, r), DIVIDE(a, b) \rightarrow (q, r, f)$. □

Although the procedures of division and gcd are partly already described in the theorems and proofs in this section we will finish this section by formalize the procedures of division, gcd and gcd$_C$, by providing the pseudocode.

Algorithm 8 Division

Input: Polynomial a and b, with variable order $\mathbf{x} = (x_1, \ldots, x_m)$
Output: Polynomials q, r with variable order $\mathbf{x} = (x_1, \ldots, x_m)$ and polynomial f with variable order $\mathbf{x} = (x_2, \ldots, x_m)$ such that $f \cdot a = q \cdot b + r$ and $deg(r) < deg(b)$

```
procedure DIVIDE($a,b$)
    if $deg(a) < deg(b)$ then
        return $(0,a,1)$
    end if
    $f_0 \leftarrow \mathbf{b}[0]$
    $\mathbf{c} \leftarrow [\,]$
    $\mathbf{c}[deg(a) - deg(b)] \leftarrow a[deg(a)]$
    $q_0 \leftarrow polynomial(c, \mathbf{x}[0])$
    $r_0 \leftarrow$ ADD(MULTIPLY($f_0, a$),MULTIPLY($q_0, b$))
    $q_1, r_1, f_1 \leftarrow$ DIVIDE($r_0, b$)
    $q, r, f \leftarrow$ ADD(MULTIPLY($f_1, q_0$) ,$q_1$),$r_1$,MULTIPLY($f_0, f_1$)
    return $(q, r, f)$
end procedure
```

REMARK In the actual implementation of division a small extra step is added. In this step all coefficients in q, r as well as the polynomial f are divided by g where

$$g = gcd(gcd_c(q), gcd_c(r), f). \tag{3.35}$$

This step is not necessary, but ensures that q, r, f do not have unnecessarily large coefficients. The reason it is not included is that adding this step most likely adds more confusion than value. □

Now we have gcd and gcd_C left to write pseudocode for. Since all the code is very recursive here, it seems to be the case that we get infinite recursive calls. This will not be the case, however, since all the time we either:

1. Decrease the degree of some polynomial

2. Decrease the size of the variable order.

The fact that one of these are true is easily checked in all the procedures.

Algorithm 9 Greatest common divisor

Input: Polynomial a and b, with variable order $\mathbf{x} = (x_1,\dots,x_m)$

Output: Polynomial g with variable order $\mathbf{x} = (x_1,\dots,x_m)$ such that g is the greatest common divisor of a and b

```
1: procedure GCD($a, b$)
2:     if $b = 0$ then
3:         return $a$
4:     end if
5:     $q_0, r_0, f_0 \leftarrow$ DIVIDE($a, b$)
6:     $g' \leftarrow$ GCD($b, r_0$)
7:     $q', r', f' \leftarrow$ DIVIDE($g'$,GCD_C($g'$))
8:     return MULTIPLY($q'$,GCD(GCD_C($a$),GCD_C($b$)))
9: end procedure
```

Algorithm 10 Greatest common coefficient divisor

Input: Polynomial a with variable order $\mathbf{x} = (x_1,\dots,x_m)$

Output: Polynomial g with variable order $\mathbf{x} = (x_2,\dots,x_m)$ such that g is the greatest common divisor of all coeffiecients a_i in a

```
1: procedure GCD_C($a$)
2:     $g \leftarrow a[0]$
3:     for $i \leftarrow 1,\dots,deg(a)$ do
4:         $g \leftarrow$ GCD($g, a[i]$)
5:     end for
6:     return $g$
7: end procedure
```

These procedure descriptions conclude a quite theory heavy section about polynomials. Next will follow Wilf-Zeilberger's method, and will decribe how the method works.

3.2 Wilf-Zeilberger's method

Wilf-Zeilberger's method is a method to show an equality by using a certifying pair, or a proof certificate. The methodology in the proof is that assume we want to prove

$$\sum_{k=-\infty}^{\infty} A(n,k) = B(n). \tag{3.36}$$

Then we let

$$F(n,k) = \frac{A(n,k)}{B(n)} \tag{3.37}$$

Now equation 3.36 becomes

$$\sum_{k=-\infty}^{\infty} F(n,k) = 1 \tag{3.38}$$

The way we want to prove this is by first proving

$$\sum_{k=-\infty}^{\infty} F(n+1,k) - F(n,k) = 0 \tag{3.39}$$

and then calculate the left hand side of 3.38 for one n, usually but not always $n = 0$. Then we get that

$$\sum_{k=-\infty}^{\infty} F(n+1,k) = \sum_{k=-\infty}^{\infty} F(n,k) \tag{3.40}$$

which in turns gives us that

$$\sum_{k=-\infty}^{\infty} F(n,k) = c, \tag{3.41}$$

where c is a constant, for all values of n and thus equal to the result of our previous calculations. The trick we use to prove equation 3.39 is to find another function $G(n,k)$ which satisfies:

i)

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k) \tag{3.42}$$

ii)

$$\lim_{k\to\pm\infty} G(n,k) = 0, \;\; \forall\, n \tag{3.43}$$

If these two conditions are fulfilled, then equation 3.39 is fulfilled since by replacing $F(n+1,k) - F(n,k)$ with $G(n,k+1) - G(n,k)$ we get a telescopic sum:

$$\begin{aligned}\sum_{k=-\infty}^{\infty} F(n+1,k) - F(n,k) &= \sum_{k=-\infty}^{\infty} G(n,k+1) - G(n,k) = \\ &= \lim_{M\to\infty} \sum_{k=-M}^{M} G(n,k+1) - G(n,k) = \\ &= \lim_{M\to\infty} \left[G(n,M+1) - G(n,-M)\right] = 0.\end{aligned}$$

If we summarize the method we have the following steps:

1. Start with equation on the form
$$\sum_{k=-\infty}^{\infty} A(n,k) = B(n).$$
2. Let
$$F(n,k) = \frac{A(n,k)}{B(n)}.$$
3. Find $G(n,k)$ such that equations 3.42 and 3.43 are fulfilled.
4. Show that $\sum_{k=-\infty}^{\infty} F(0,k) = 1$.

We can formalize the concepts in the method like this.

DEFINITION 3.9 A pair of functions (F,G) that satisfy equations 3.42 and 3.43 are said to "certify" an identity like 3.38 (or is simply called a "certifying pair"). We can also speak of a "proof certificate" $R(n,k)$ of an identity. That is a function such that $G(n,k) = R(n,k)F(n,k-1)$ gives that (F,G) is a certifying pair to the identity. $\square$

Let us show an short example of how the method can work.

EXAMPLE 3.5 Show that
$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

SOLUTION We will use the method and do all the steps mentioned above.

1. We have $A(n,k) = \binom{n}{k}$ and $B(n) = 2^n$.
2. Let
$$F(n,k) = \frac{A(n,k)}{B(n)} = \frac{\binom{n}{k}}{2^n}.$$
3. With
$$G(n,k) = -\frac{\binom{n}{k-1}}{2^{n+1}},$$

we have that

$$\begin{aligned}F(n+1,k)-F(n,k) &= \frac{\binom{n+1}{k}}{2^{n+1}} - \frac{\binom{n}{k}}{2^{n}} = \\ &= \frac{1}{2^{n+1}}\binom{n}{k}\left(\frac{n+1}{n+1-k}-2\right) = \\ &= \frac{1}{2^{n+1}}\binom{n}{k}\left(\frac{k}{n+1-k}-1\right) = \\ &= \frac{1}{2^{n+1}}\left(\binom{n}{k-1}-\binom{n}{k}\right) = \\ &= G(n,k+1)-G(n,k)\end{aligned}$$

Furthermore, we have that $G(n,k) = 0$ when $k < 0$ and $k > n$.

4. Lastly we need to show that

$$\sum_{k=0}^{n} \frac{\binom{n}{k}}{2^n} = 1$$

for some n. If we choose $n = 0$ we see that this equality holds. Therefore the equality holds for all n. □

Now that we have seen an example, hopefully the methodology is clear. What still is not clear is how to find the function $G(n,k)$. This is in general a hard task to do by hand, but is certainly managable by using a computer since there are algorithms – for instance Gosper's algorithm – that can derive $G(n,k)$ in certain cases. This implementation is what this thesis is contributing to do.

3.3 Gosper's algorithm

Gosper's algorithm is an algorithm that can be used to find an sum when certain conditions are fulfilled for the sum. The setting of the problem that is solved using the algorithm is that we want to find S_k where

$$\sum_{k=1}^{n} a_k = S_n - S_0. \tag{3.44}$$

This is the same thing as finding S_n such that

$$a_k = S_k - S_{k-1}. \tag{3.45}$$

The algorithm provides those S_k such that

$$\frac{S_k}{S_{k-1}} = \text{rational function of } k. \tag{3.46}$$

Note that all the time when we write x_a in this section, this denotes a polynomial x evaluated in a, or $x(a)$. For instance S_k denotes that the polynomial $S(k)$. We are only using this notation to make it easier to read. Note that this means that x is a function of the variable a, but does not mean that it is a one variable function of a – it can be a function of several variables but we are just interested in the variable a while the others are viewed as constants. This will actually be the cast amost all the time, since $a_k = F(n+1,k) - F(n,k)$ (where F comes from Wilf-Zeilberger's method) will be used most of the time. Also here, note that $F(n,k)$ only denotes that F is a function of n and k, but it might be of more variables as well.

For the sake of making the thesis more easily readable and for completeness, we will provide a quite thorough derivation of the algorithm – even though it is perfectly described in the original paper by Gosper (1978). This will mostly focus on the steps of the algorithm, but also include the proofs of claims that are made during the steps. Firstly we will describe the main steps and thereafter we will provide the proofs.

First of all we need to show the connection between the formulation of Wilf-Zeilberger's method and how Gosper's algorithm takes part in it. Gosper's algorithm will solve the third step in Wilf-Zeilbergers method, namely to find a $G(n,k)$ such that conditions 3.42 and 3.43 will be fulfilled. The first condition is that

$$F(n+1,k) - F(n,k) = G(n,k+1) - G(n,k). \tag{3.47}$$

By looking at n as a constant we define $a_k = F(n+1,k) - F(n,k)$. Thereafter we use Gosper's algorithm to find a S_k such that

$$a_k = S_k - S_{k-1}. \tag{3.48}$$

Once we have that, we let $G(n,k) = S_{k-1}$ and we have obtained our function $G(n,k)$.

The steps of Gosper's algorithm are:

1. Finding polynomials p_k, q_k, r_k such that

$$\frac{a_k}{a_{k-1}} = \frac{p_k}{p_{k-1}} \frac{q_k}{r_k} \tag{3.49}$$

   and $\gcd(q_k, r_{k+j}) = 1, \ \forall\ j \geq 0$.

2. Finding polynomial f_k such that

$$p_k = q_{k+1} f_k - r_k f_{k-1}. \tag{3.50}$$

3. Let

$$S_k = \frac{q_{k+1}}{p_k} f_k a_k. \tag{3.51}$$

The reason this algorithm will provide a solution is that with S_k given by 3.51 we have

$$S_k - S_{k-1} = \frac{q_{k+1}}{p_k} f_k a_k - \frac{q_k}{p_{k-1}} f_{k-1} a_{k-1}. \tag{3.52}$$

By factoring out $\frac{a_k}{p_k}$ in 3.52, and using 3.49 followed by 3.50 gives us:

$$\begin{aligned} S_k - S_{k-1} &= \frac{a_k}{p_k}\Big(q_{k+1} f_k - \frac{q_k}{p_{k-1}} f_{k-1} p_k \frac{a_{k-1}}{a_k}\Big) = \\ &= \frac{a_k}{p_k}\Big(q_{k+1} f_k - \frac{q_k}{p_{k-1}} f_{k-1} p_k \frac{p_{k-1}}{p_k}\frac{r_k}{q_k}\Big) = \\ &= \frac{a_k}{p_k}\Big(q_{k+1} f_k - r_k f_{k-1}\Big) = \frac{a_k}{p_k} p_k = a_k, \end{aligned} \tag{3.53}$$

which is exactly what we wanted to be true for S_k.

Now, we have quite a few details to derive in each step – both how the step is done more precisely and furthermore explaining why all the polynomials actually need to be polynomials and not rational polynomials instead.

3.3.1 How to get polynomials p, q, r in equation 3.49

First of all, we need to prove that it is possible to find polynomials p, q, r such that 3.49 is fulfilled. Since $\frac{S_k}{S_{k-1}}$ is a rational function of k, then

$$\frac{a_k}{a_{k-1}} = \frac{S_k - S_{k-1}}{S_{k-1} - S_{k-2}} = \frac{\frac{S_k}{S_{k-1}} - 1}{1 - \frac{S_{k-2}}{S_{k-1}}} \tag{3.54}$$

is a rational function of k as well. Therefore we will be able to find polynomials such that 3.49 is fulfilled. Left is to show how to find polynomials such that $gcd(q_k, r_{k+j}) = 1 \ \ \forall\ j \geq 1$ as well.

We do this in a series of steps. First we let $p_k = 1$ and q_k, r_k be the numerator and denominator of $\frac{a_k}{a_{k-1}}$, respectively. Then if $gcd(q_k, r_{k+j}) = 1, \ \ \forall\ j \geq 0$ we are done.

Otherwise we replace the polynomials by

$$\begin{aligned} q'_k &\leftarrow \frac{q_k}{g_k}, \\ r'_k &\leftarrow \frac{r_k}{g_{k-j}}, \\ p'_k &\leftarrow p_k g_k g_{k-1} \dots g_{k-j+1}, \end{aligned} \tag{3.55}$$

where $g_k = gcd(q_k, r_{k+j})$ for the smallest $j \geq 0$ such that $gcd(q_k, r_{k+j}) \neq 1$. Then we can see that we still have that

$$\frac{a_k}{a_{k-1}} = \frac{p'_k}{p'_{k-1}} \frac{q'_k}{r'_k} \tag{3.56}$$

and the degree of the polynomials q_k, r_k are smaller. We then iterate this procedure as long as there exists a j such that $gcd(q_k, r_{k+j}) > 1$. Then when this procedure finishes, we will obtain q_k, r_k, p_k such that 3.49 is fulfilled and $gcd(q_k, r_{k+j}) = 1 \;\; \forall\, j \geq 0$.

3.3.2 How to get polynomial f in equation 3.50

The proof provided here is just due to Gosper's paper and mostly very similar. First of all, we need to prove that f is a polynomial and not a rational polynomial.

Proof Assume that $f_k = \frac{c_k}{d_k}$ where $gcd(c_k, d_k) = 1$. Then by plugging this into 3.50 and multiplying by $d_k d_{k-1}$ gives us:

$$d_k d_{k-1} p_k = c_k d_{k-1} q_{k+1} - d_k c_{k-1} r_k. \tag{3.57}$$

Let j be the largest integer such that

$$gcd(d_k, d_{k+j}) = g_k \neq 1 \tag{3.58}$$

Since $g_k | d_{k+j}$ we get that

$$gcd(d_{k-1}, d_{k+j}) = 1 = gcd(d_{k-1}, g_{k+j}). \tag{3.59}$$

By just shifting k by $-j-1$ in 3.58 we get that

$$gcd(d_{k-j-1}, d_{k-1}) = g_{k-j-1} \neq 1 \tag{3.60}$$

By shifting k by j in 3.59 we get that

$$gcd(d_{k-j-1}, d_k) = 1 = gcd(g_{k-j-1}, d_k), \tag{3.61}$$

again since $g_{k-j-1} | d_{k-j-1}$.

Now we consider equation 3.57 upon dividing by first g_k and then g_{k-j-1}. Clearly $g_k | d_k d_{k-1} p_k$, since $g_k | d_k$. This means that

$$g_k | c_k d_{k-1} q_{k+1} - d_k c_{k-1} r_k. \tag{3.62}$$

Furthermore $g_k | d_k c_{k-1} r_k$ since $g_k | d_k$, which gives us that

$$g_k | c_k d_{k-1} q_{k+1}. \tag{3.63}$$

By equation 3.59 we get $gcd(d_{k-1}, g_k) = 1$ and since $g_k | d_k$ and $gcd(c_k, d_k) = 1$ we get $gcd(c_k, g_k) = 1$. This means that

$$g_k | q_{k+1} \implies g_{k-1} | q_k. \tag{3.64}$$

Similarly $g_{k-j-1} | d_k d_{k-1} p_k$, since $g_{k-j-1} | d_{k-1}$. This means that

$$g_{k-j-1} | c_k d_{k-1} q_{k+1} - d_k c_{k-1} r_k. \tag{3.65}$$

Furthermore $g_{k-j-1} | c_k d_{k-1} q_{k+1}$ since $g_{k-j-1} | d_{k-1}$, which gives us that

$$g_k | d_k c_{k-1} r_k. \tag{3.66}$$

By equation 3.61 we get $gcd(d_k, g_{k-j-1}) = 1$ and by $gcd(c_{k-1}, d_{k-1}) = 1$ together with $g_{k-j-1} | d_{k-1}$ we get $gcd(c_{k-1}, g_{k-j-1}) = 1$. This means that

$$g_{k-j-1} | r_k \implies g_{k-1} | r_{k+j}. \tag{3.67}$$

But now we have g_{k-1} divides both r_{k+j} where $j \geq 0$ and q_k. From the first step of Gosper's algorithm we know that $gcd(r_{k+j}, q_k) = 1, \ \ \forall\ j \geq 0$ meaning that $g_k = 1$. This means that for all $j \geq 0$ we have that

$$gcd(d_k, d_{k+j}) = 1. \tag{3.68}$$

By putting $j = 0$ we get that

$$d_k = gcd(d_k, d_k) = 1. \tag{3.69}$$

Hence $d_k = 1$ for all k and thus f_k is a polynomial. □

Now, let us derive how to get the polynomial f_k such that 3.50 is fulfilled. It is possible to get a bound for the degree of f_k with respect to k. This is derived in the paper by Gosper (1978) and for the interested reader the details can be found there. The results is that

$$deg(f_k) \leq deg(p_k) - max(deg(q_{k+1} + r_k), deg(q_{k+1} - r_k)) + 1. \tag{3.70}$$

In some cases the degree can be further limited (without the extra $+1$) but we are mostly interested in that we have a bound – not the extra added (small) constant.

We will now firstly show that there is a way to get the polynomial f_k given that the degree of f_k is less than N for all variables in f_k. This is done by assigning

$$f_k = \sum_{\substack{\mathbf{i}\in(0,1,\ldots,N)^{m-1}\\ i_1\in(0,1,\ldots,N)}} \left(a_{\mathbf{i},i_1} k^{i_1} \prod_{j=2}^{m} v_j^{i_j}\right), \tag{3.71}$$

where m is the number of variables in f_k, the variables of f_k are named v_j for $1 \leq j \leq m$ where $v_1 = k$, $a_{\mathbf{i}}$ are the coefficients and $\mathbf{i} = (i_2, i_3, \ldots, i_m)$ goes through all combinations of degrees in the range $0 \leq deg(v_j) = i_j \leq N$ for all the different variables. The number i_1 is the exponent of k and also goes through all the values for in the range $0 \leq i_1 \leq N$. This exponent is treaded slightly different just because it is the exponent of k.

Then from this we can derive what the polynomial f_{k-1} is. First we write the polynomial f_{k-1} on the same form as in 3.71 but with other coefficients:

$$f_{k-1} = \sum_{\mathbf{i}\in(0,1,\ldots,N)^{m}} \left(b_{\mathbf{i},i_1} k^{i_1} \prod_{j=2}^{m} v_j^{i_j}\right). \tag{3.72}$$

Now we will derive the expression for $b_{\mathbf{i},i_1}$ in terms of $a_{\mathbf{i},i_1}$. If we replace k by $k-1$ in 3.71 we get

$$\begin{aligned} f_k &= \sum_{\substack{\mathbf{i}\in(0,1,\ldots,N)^{m-1}\\ i_1\in(0,1,\ldots,N)}} \left(a_{\mathbf{i},i_1} (k-1)^{i_1} \prod_{j=2}^{m} v_j^{i_j}\right) = \\ &= \sum_{\substack{\mathbf{i}\in(0,1,\ldots,N)^{m-1}\\ i_1\in(0,1,\ldots,N)}} \left(a_{\mathbf{i},i_1} \left(\sum_{u=0}^{i_1} \binom{i_1}{u} k^u (-1)^{i_1-u}\right) \prod_{j=2}^{m} v_j^{i_j}\right) = \\ &= \sum_{\substack{\mathbf{i}\in(0,1,\ldots,N)^{m-1}\\ i_1\in(0,1,\ldots,N)}} \left(\sum_{u=0}^{i_1} a_{\mathbf{i},i_1} \binom{i_1}{u} (-1)^{i_1-u} k^u\right) \prod_{j=2}^{m} v_j^{i_j} \end{aligned} \tag{3.73}$$

Now we can see that the term $k^{i'_1} \prod_{j=2}^{m} v_j^{i_j}$ from 3.72 (i'_1 denotes a specific i_1) appears in 3.73 in several places – namely when $i_1 \in \{i'_1, i'_1+1, \ldots, N\}$. This means that we get

$$b_{\mathbf{i},i_1} = \sum_{s=i_1}^{N} a_{\mathbf{i},s} \binom{s}{i_1} (-1)^{s-i_1} \tag{3.74}$$

Now we have all the background we need in order to set up the system of equations and then solve for $a_{\mathbf{i},i_1}$. This is done by noticing that the left and right hand sides of the equation

$$p_k = q_{k+1} f_k - r_k f_{k-1} \tag{3.75}$$

In order to ease the formulation of the equation system, let $\hat{\mathbf{i}} = (\mathbf{i}, i_1)$, let $\ll$ denote that a vector is pointwise smaller than or equal to e.g. $(1,2,3) \ll (2,3,3)$ and let $\mathbf{i}-\mathbf{j}$ denote the pointwise difference between $\mathbf{i}$ and $\mathbf{j}$, e.g. $(2,3,3)-(1,2,3) = (1,1,0)$. Finally we also denote the coefficients in front of $k^{i_1}\prod_{j=2}^{N} v_j^{i_j}$ in the polynomials p_k, q_{k+1}, r_k by $r_{\hat{\mathbf{i}}}, q_{\hat{\mathbf{i}}}, r_{\hat{\mathbf{i}}}$, respectively. This gives us that

$$p_{\hat{\mathbf{i}}} = \sum_{\substack{\hat{\mathbf{j}} \\ \hat{\mathbf{j}} \ll \hat{\mathbf{i}} \\ \hat{\mathbf{j}}' = \hat{\mathbf{i}} - \hat{\mathbf{j}}}} a_{\hat{\mathbf{j}}} q_{\hat{\mathbf{j}}'} - \sum_{\substack{\hat{\mathbf{j}} \\ \hat{\mathbf{j}} \ll \hat{\mathbf{i}} \\ \hat{\mathbf{j}}' = \hat{\mathbf{i}} - \hat{\mathbf{j}}}} b_{\hat{\mathbf{j}}} r_{\hat{\mathbf{j}}'}, \tag{3.76}$$

where $b_{\hat{\mathbf{j}}}$ is given by 3.74. Note that at this point in the algorithm all the numbers $p_{\hat{\mathbf{i}}}, q_{\hat{\mathbf{i}}}, r_{\hat{\mathbf{i}}}$ are simply constants. This means that we can formulate a system of equations

$$Ma = P, \tag{3.77}$$

where M is the matrix that is given by the right hand side of 3.76, a consists of all coefficients $a_{\hat{\mathbf{j}}}$ and P is the left hand side of 3.76. Note that each row in this system of equations correspond to a different $\hat{\mathbf{i}}$. Here we go through all $\hat{\mathbf{i}}$ that are relevant – meaning all $\hat{\mathbf{i}}$ with degree smaller than what either of the sides of equation 3.76 can have. This means that we have an overdetermined system of equations. Furthermore we demand that all the coefficients of f_k have to be integers. This means that it is not certain that we will get a solution, but in case we do we have found f_k. If we do not get a solution that can be because of two different reasons:

i) We chose a too small maximal degree N for the polynomial f_k.

ii) There does not exist a solution, hence the identity cannot be shown by Wilf-Zeilbergers method.

In the paper by Gosper (1978) the first reason for not finding a f_k does not exists, since the paper proves an upper limit on the degree of f_k with respect to k. This cannot be done for other variables, for instance with $p_k = q_{k+1} = r_k = 1$ any polynomial f_k on the form $f_k = n^c + k$ satisfies 3.75.

Is it possible to show if there exists a solution there exist one with limited degree

In the system 3.77, M can be constructed to have dimension $L \times (N^m)$, where $L = max(deg(p_k)+1, max(deg(q_{k+1}), deg(r_k)) + N + 1)$. Here $deg(g)$ denotes the largest degree in any variable of the polynomial g. The reason for this limit is that the row with highest degree for any variable can be chosen to be L, since the degree of either side of the equation 3.76 is at most L.

The time complexity for this system of equations is $\mathcal{O}(L \cdot N^{2m})$, since we perform the following algorithm:

Algorithm 11 Gaussian Elimination

Input: Matrix M, vector P
Output: Vector x such that $Mx = P$

```
 1: procedure GaussianElimination($M, P$)
 2:     $r \leftarrow$ Number of rows in M
 3:     $c \leftarrow$ Number of columns in M
 4:     for $row \leftarrow 1, \ldots, c$ do
 5:         $i \leftarrow$ first row with row number $\geq row$ and $M[i][c] \neq 0$
 6:         if No such $i$ exists then
 7:             Continue loop
 8:         end if
 9:         $swap(M[row], M[i])$
10:         $swap(P[row], P[i])$
11:         for $row_2 \leftarrow row + 1, \ldots, L$ do
12:             $g \leftarrow gcd(M[row][c], M[row_2][c])$
13:             $M[row_2] \leftarrow M[row_2] \cdot (M[row][c]/g) - M[row] \cdot (M[row_2][c]/g)$
14:             $P[row_2] \leftarrow P[row_2] \cdot (M[row][c]/g) - P[row] \cdot (M[row_2][c]/g)$
15:         end for
16:     end for
17: end procedure
```

Remark In the code we multiply a row vector by an integer. This means that we perform elementwise multiplication with this integer. □

Here we can clearly see that we have two loops, which are of size $c = N^m$ and $r = L$, respectively. The reason for the last N^m-factor is that when we perform row 9–14 we do this elementwise, hence once for each column.

This means that the time complexity is exponential in the number of variables. In general exponential time complexities are not desired at all – since they grow very fast. In this thesis however we are working with identities that just involve a few (less than 5) different variables. Therefore this time complexity is acceptable. This means that given that given the limit on m we have a polynomial time complexity for obtaining f_k.

4

Implementation

4.1 Preprocessing of input

4.2 Gosper's algorithm

4.2.1 Finding p,q,r

4.2.2 Finding f

4.3 Proof generation

5 Results

6

Discussion and conclusions

References

Attachments

6.1 Code

Popular scientific paper